Oracle® Telecommunications Service Ordering

Process Guide

Release 11*i*

Part No. B10674-01

August 2003

ORACLE®

Oracle Telecommunications Service Ordering Process Guide, Release 11*i*

Part No. B10674-01

Contents

Send Us Your Comments ix

Preface xi

Part I Overview

1 Introduction

1.1 Purpose 1-2

2 TSO Business Process

2.1 Telecommunications Services Ordering Business Process 2-2
2.1.1 Create a Quote 2-3
2.1.2 Configure New Items 2-5
2.1.3 Reconfigure a Customer's Installed Services 2-6
2.1.4 Convert the Quote to a Sales Order and Verify Order Fulfillment 2-8

Part II Functionality and Setup

3 Process Functionality

3.1 Key Functionality of the TSO Solution 3-2
3.2 Quoting 3-3
3.3 Pricing 3-5
3.4 Configurator 3-7
3.5 Install Base 3-11
3.6 Service Fulfillment Manager 3-12

4 Setup Flow and Checklist

4.1 About Setting Up Telecommunications Service Ordering 4-2

4.2 Setup Flow 4-2
4.3 Setup Checklist 4-4

5 Set Up Order Management

5.1 Ensure Application of Patch 2529922 5-3
5.2 Modification to OM Workflow Processes 5-3
5.2.1 Create SFM Enabled Order Header and Line Workflows 5-3
5.2.2 Replace the Fulfill Node in the OM Line Workflow 5-3
5.3 Set Up Transaction Types 5-4
5.3.1 Set Up Line Transaction Type 5-4
5.3.2 Set Up Order Transaction Type 5-5
5.4 Design Tips 5-6

6 Set Up Inventory

6.1 Setup of Telecommunications Service Items 6-3
6.2 Specify Install Base Tracking for Each Component Item 6-3
6.3 Set an Item as Provisionable 6-4
6.4 Set Up an Item of Type Link 6-5
6.5 Create a Container Model 6-6
6.5.1 Specify Container Model 6-6

7 Set Up Bill of Materials

7.1 About Setting Up BOM 7-2

8 Set Up Install Base

8.1 Enable Network Configurations 8-3
8.2 Create Extended Attributes 8-3
8.2.1 Set Up Extended Attribute Pools 8-4
8.3 Map Extended Attributes to Items 8-6
8.4 Design Tips 8-7

9 Set Up Advanced Pricing

9.1 Set Up One-Time Charges 9-3

9.1.1 Create One-Time Charge Names ... 9-3
9.1.2 Define Modifier for an Item or Service ... 9-4
9.1.3 Define Qualifiers on the Modifier ... 9-7
9.2 Discount to One-Time Charges ... 9-8
9.3 Pricing Attributes and Sourcing Rules ... 9-9
9.4 Get Custom Price ... 9-13

10 Set Up Configurator and Customize the Solution

10.1 About Configurator Implementation in the Solution ... 10-3
10.1.1 Conventions for Connector and Components of a Model ... 10-3
10.1.2 Using Container Models ... 10-4
10.1.3 Container Model Settings and Structure ... 10-4
10.1.4 Importing Container Models into Oracle Configurator ... 10-5
10.1.5 Structuring Container Models ... 10-7
10.1.6 Connecting Components ... 10-11
10.1.7 Using Configuration Rules ... 10-13
10.2 Create the Configuration Model ... 10-16
10.3 Publish a Container Model ... 10-18
10.4 Define Profile Options ... 10-18
10.5 Disable Pricing ... 10-19
10.6 CZ Schema Customizations ... 10-19
10.6.1 TSO Solution CZ_DB_SETTINGS Parameters ... 10-20
10.6.1.1 DISPLAY_SUMMARY_FULFILLMENT_ACTION ... 10-20
10.6.2 CZ_CONFIG_ATTRIBUTES Table ... 10-21
10.6.3 CZ_CONFIG_EXT_ATTRIBUTES Table ... 10-21
10.6.4 Map Install Base Extended Attributes to CZ Attributes in Developer ... 10-22
10.6.5 MACD_FULFILLMENT_STRUCTURES Table ... 10-23
10.6.6 Mapping of CZ tables to Install Base Schema (CSI) ... 10-25
10.7 Initialization Parameters ... 10-27
10.7.1 instance ... 10-28
10.7.2 validation_context ... 10-28
10.8 Batch Validation Parameters ... 10-29
10.9 Programmatic Tools for TSO Development ... 10-30
10.9.1 COPY_CONFIGURATION ... 10-31
10.9.1.1 Considerations Before Running ... 10-32

10.9.1.2 Syntax and Parameters 10-32
10.9.1.3 Considerations After Running 10-34
10.9.2 COPY_CONFIGURATION_AUTO 10-35
10.9.3 VALIDATE 10-35
10.9.3.1 Syntax and Parameters 10-35

11 Set Up Configurator Functional Companions

11.1 Modifying the TSO Functional Companions 11-3
11.1.1 Overview 11-3
11.1.2 Specify the Default Location 11-5
11.1.2.1 Purpose of the Default Location Functional Companion 11-5
11.1.2.2 Example Code for the Default Location Functional Companion 11-5
11.1.2.3 Setup for the Default Location Functional Companion 11-6
11.1.2.4 Effects of the Default Location Functional Companion 11-7
11.1.2.5 Modify the Default Location Functional Companion 11-7
11.1.3 Specify the Line Type 11-9
11.1.3.1 Purpose of the Line Type Functional Companion 11-9
11.1.3.2 Example Code for the Line Type Functional Companion 11-9
11.1.3.3 Setup for the Line Type Functional Companion 11-10
11.1.3.4 Effects of the Line Type Functional Companion 11-14
11.1.3.5 Modify the Line Type Functional Companion 11-18
11.1.4 Change the Instance Name 11-18
11.1.4.1 Purpose of the Instance Name Functional Companion 11-18
11.1.4.2 Example Code for the Instance Name Functional Companion 11-18
11.1.4.3 Setup for the Instance Name Functional Companion 11-18
11.1.4.4 Effects of the Instance Name Functional Companion 11-20
11.1.4.5 Modify the Instance Name Functional Companion 11-21
11.1.5 Using Configuration Attributes with Install Base (IB) 11-21
11.1.5.1 Purpose of the IBAttribute Functional Companion 11-21
11.1.5.2 Example Code for the IBAttribute Functional Companion 11-22
11.1.5.3 Setup for the IBAttribute Functional Companion 11-22
11.1.5.4 Effects of the IBAttribute Functional Companion 11-23
11.1.5.5 Modify the IBAttribute Functional Companion 11-23
11.2 Using the Oracle Configuration Interface Object 11-25
11.2.1 Update Installed Configurations 11-25

11.2.1.1 Identify Changes to a Configuration ... 11-25
11.2.1.2 Accessing Discontinued Items ... 11-27
11.2.1.3 Accessing Instances ... 11-27
11.2.1.4 Determining Editability ... 11-28
11.2.2 Handling Interface Events ... 11-28
11.2.3 Automatic Behavior for Configurations ... 11-31

12 Set Up Quoting

12.1 Summary of Reconfiguration ... 12-2
12.2 Define Profile Options ... 12-3
12.3 Set Up the Lines Page ... 12-6
12.3.1 Display or Hide Columns ... 12-8
12.3.2 Hide a Seeded Column ... 12-10
12.3.3 Edit Column Labels ... 12-11
12.3.4 Edit Column Sequence ... 12-12
12.4 Enable or Disable Line-Level Actions or Table-Level Buttons ... 12-13
12.5 Change Default Number of Rows ... 12-15
12.6 Design Tips ... 12-16

13 Set Up Service Fulfillment Manager

13.1 Create SFM Enabled Order Management Workflow Header Process ... 13-3
13.2 Create SFM Enabled Order Management Workflow Line Process ... 13-4
13.3 Add Function for Menu Notifications ... 13-5
13.4 Create a PL/SQL Procedure That Updates a Certain Workitem Parameter ... 13-6
13.5 Create Workflow Process ... 13-7
13.6 Define Work Item ... 13-9
13.7 Map Work Item to Item and Action (Line Transaction Type) Combination ... 13-10
13.8 Design Tips ... 13-11

Part III Using and Administering the Solution

14 Using the Solution

14.1 Introduction ... 14-2
14.2 Create a New Quote ... 14-2

14.3 Add Products to the Quote 14-4
14.4 Reconfigure an Existing Service 14-5
14.5 Add Products to Quote from Install Base 14-6
14.6 Add to Container Model from Install Base 14-7
14.7 Remove Unchanged Components 14-8
14.8 Remove Lines 14-9
14.9 Place an Order 14-10

15 Administering the Solution

15.1 Re-validate Installed Configurations After Modifications 15-2
15.2 Managing Notification Errors 15-3
15.3 Find an Order in the SFM Order Flow-Through Area 15-4
15.4 Retry Install Base Updates 15-4
15.5 Retry Failed Outgoing Messages 15-5
15.6 Manage Failure Notifications in HTML 15-7

Part IV Appendixes

A Assumptions and Restrictions

A.1 List of Assumptions and Restrictions A-1

B Functional Companion Code Examples

B.1 Setting the Default Location B-3
B.2 Setting the Line Type B-5
B.3 Using Configuration Attributes with Install Base (IB) B-12
B.4 Changing the Instance Name B-22
B.5 Determining Editability B-24

Glossary

Index

Send Us Your Comments

Oracle Telecommunications Service Ordering Process Guide, Release 11*i*

Part No. B10674-01

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 way:

- Electronic mail: mfgdoccomments_us@oracle.com

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

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

Preface

Welcome to the *Oracle Telecommunications Service Ordering Process Guide*.

This document provides the reader with an understanding of the TSO solution, the overall user flow and functionality that the solution supports, and setup information to implement this solution.

Intended Audiences

This document is intended for implementers and end users of a TSO solution. Before using this book as an implementer, you should have a working knowledge of Oracle Order Management, Oracle Inventory, Bills of Material, Oracle Install Base, Oracle Configurator, Oracle Quoting, and Oracle Service Fulfillment Manager. For guidance in determining which parts of this book are of interest to you, see the Structure section on page xiii.

Audience and User Definitions

This document has multiple audiences and refers to different audiences, users, and customers. These audiences and users include:

- **Implementer**: The technically oriented person who thoroughly understands setting up a variety of Oracle applications and who is setting up the TSO solution for purchaser of the TSO solution. A key area of interest to the implementer is Part II, "Functionality and Setup", which provides the implementer technical overviews and solution-specific setup information.
- **End user**: The person who interacts with directly customers who bought or are buying telecommunication services or products. This person might be a customer service representative who is helping their customer with specifying a

sales order. A key area of interest to the end user is Chapter 14, "Using the Solution".

- **Administration user**: The person who monitors and maintains sales orders after placement of the sales orders. The tasks that this person performs are "behind the scene" and do not involve direct contact with the customer. A key area of interest to the administration user is Chapter 15, "Administering the Solution".
- **Customer**: The person who is purchasing or is changing their telecommunication services or products. This person is **not** an intended audience for this document and would typically have no knowledge of Oracle Applications or the TSO solution.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

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

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

The *Oracle Telecommunications Service Ordering Process Guide* comprises:

Part I, "Overview"

This part introduces the TSO solution.

Chapter 1, "Introduction"

- This chapter includes a high-level summary of the Telecommunications Service Ordering (TSO) solution and a list of terms and definitions.

Chapter 2, "TSO Business Process"

- This chapter describes business flows in the Telecommunications Service Ordering solution.

Part II, "Functionality and Setup"

This part explains the functionality of the TSO solution and how to set up the TSO process.

Chapter 3, "Process Functionality"

- This chapter describes solution functionality by Oracle Application in the Telecommunications Service Ordering solution.

Chapter 4, "Setup Flow and Checklist"

- This chapter provides you a suggested setup flow and checklist.

Chapter 5, "Set Up Order Management"

- This chapter provides you setup information for Oracle Order Management that is specific to the Telecommunications Service Ordering solution.

Chapter 6, "Set Up Inventory"

- This chapter provides you setup information for Oracle Inventory that is specific to the Telecommunications Service Ordering solution.

Chapter 7, "Set Up Bill of Materials"

- This chapter provides you setup information for Oracle Bill of Materials that is specific to the Telecommunications Service Ordering solution.

Chapter 8, "Set Up Install Base"

- This chapter provides you setup information for Oracle Install Base that is specific to the Telecommunications Service Ordering solution.

Chapter 9, "Set Up Advanced Pricing"

- This chapter provides you setup information for Oracle Advanced Pricing that is specific to the Telecommunications Service Ordering solution

Chapter 10, "Set Up Configurator and Customize the Solution"

- This chapter provides you setup information for Oracle Configurator that is specific to the Telecommunications Service Ordering solution.

Chapter 11, "Set Up Configurator Functional Companions"

- This chapter provides you setup information for Oracle Configurator Functional Companions that is specific to the Telecommunications Service Ordering solution.

Chapter 12, "Set Up Quoting"

- This chapter provides you setup information for Oracle Quoting that is specific to the Telecommunications Service Ordering solution.

Chapter 13, "Set Up Service Fulfillment Manager"

- This chapter provides you setup information for Oracle Service Fulfillment Manager that is specific to the Telecommunications Service Ordering solution.

Part III, "Using and Administering the Solution"

This part explains how to use the TSO solution and perform administration tasks.

Chapter 14, "Using the Solution"

- This chapter describes the end-user procedures of a TSO solution.

Chapter 15, "Administering the Solution"

- This chapter provides procedural information that an administrator performs to monitor sales orders to check for and respond to failure notification messages.

Part IV, "Appendixes"

This part provides you with extra information about which you should be aware and examples of code that Functional Companion uses.

Appendix A, "Assumptions and Restrictions"

- This chapter provides you with various assumptions and restrictions of which you should be aware.

Appendix B, "Functional Companion Code Examples"

- This chapter provides fuller and longer examples that the Configurator setup chapters use.

Glossary

This glossary contains definitions that you may need while implementing a Telecommunications Services Ordering solution.

Index

The Index lets you search for contents by key words.

Related Documents

The following documents provide Oracle Application-specific information relating to the Oracle Telecommunications Service Ordering solution.

- *Oracle Advanced Pricing Implementation Manual*
- *Oracle Advanced Pricing User's Guide*
- *Oracle Inventory User's Guide*
- *Oracle Bills of Material User's Guide*
- *Oracle Install Base Concepts and Procedures*
- *Oracle Install Base Implementation Guide*
- *Oracle Order Management Suite Implementation Manual*
- *Oracle Order Management User's Guide*
- *Oracle Configurator Developer User's Guide*
- *Oracle Configurator Implementation Guide*
- *Oracle Configurator Installation Guide*
- *Oracle Configuration Interface Object (CIO) Developer's Guide.*
- *Oracle Configurator Methodologies*
- *Oracle Configurator Release Notes*

- *Oracle Quoting Implementation Guide*
- *Oracle Quoting User Guide*
- *Oracle Service Fulfillment Manager Implementation Guide*

Conventions

The following conventions are used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
italic	Italic type in text, tables, or code examples indicates user-supplied text. Replace these placeholders with a specific value or string.
`name()`	In text other than code examples, the names of programming language methods and functions are shown with trailing parentheses. The parentheses are always shown as empty. For the actual argument or parameter list, see the reference documentation. This convention is not used in code examples.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Part I

Overview

This part contains the following chapters:

- Chapter 1, "Introduction"
- Chapter 2, "TSO Business Process"

1 Introduction

The topic in this chapter includes:

- Section 1.1, "Purpose" on page 1-2

1.1 Purpose

Ordering and delivering telecommunication services is complex. The complexity increases significantly with service changes. The TSO functionality in Oracle Applications supports the processes involved in moving, adding, changing, and disconnecting customer services.

Oracle Applications reduce the complexity of the ordering process by providing:

- A centralized view of the customer, including the installed service configuration (Oracle Install Base).
- The products and services available to the customer.
- The ability to restore, reconfigure, and re-price an existing service configuration.

The TSO solution spans multiple Oracle applications, including:

- Oracle Order Management
- Oracle Inventory
- Oracle Bills of Material
- Oracle Install Base
- Oracle Configurator
- Oracle Quoting
- Oracle Service Fulfillment Manager

Each application has developed a set of functionality which when used together with all the other applications enables the TSO solution. Section 3.1, "Key Functionality of the TSO Solution" on page 3-2 provides you with a summary of TSO functionality by Oracle application.

The Oracle Telecommunications Service Ordering (TSO) solution enables you to update existing configurations of telecommunication services by moving, adding, changing, or disconnecting a customer's services.To enable the upgrade process flow, companies need the ability to track the full life cycle of an item instance--product information, location, status, and so on--from order creation to order fulfillment. If a customer requests an upgrade of a product from a company after several years have passed, the company must know the exact current state of that product and must verify that any requested changes are feasible, such as whether or not these changes are compatible with a customer's existing installed product, and fulfilling only those requested changes.

2

TSO Business Process

Main topics in this chapter include:

- Section 2.1, "Telecommunications Services Ordering Business Process" on page 2-2
- Section 2.1.1, "Create a Quote" on page 2-3
- Section 2.1.2, "Configure New Items" on page 2-5
- Section 2.1.3, "Reconfigure a Customer's Installed Services" on page 2-6
- Section 2.1.4, "Convert the Quote to a Sales Order and Verify Order Fulfillment" on page 2-8

About this Chapter

This chapter provides you with high-level flow descriptions of end-user TSO business processes. If you would like procedural directions that the end-user performs, see Chapter 14, "Using the Solution".

Alternately, if you would like information about TSO setup and processes, see Chapter 4, "Setup Flow and Checklist".

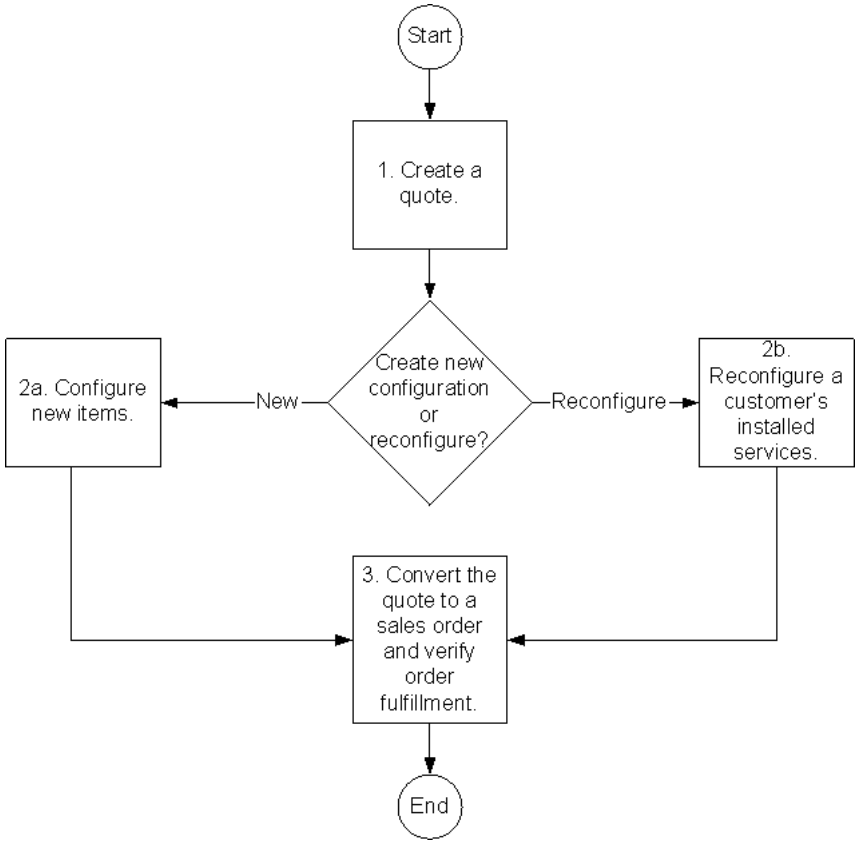
2.1 Telecommunications Services Ordering Business Process

The Telecommunications Services Ordering solution consists of the follow processes.

- Creating a new configuration for a customer
- Reconfiguring a customer's installed services

Figure 2–1, "Overall TSO Business Process" shows the overall Telecommunications Services Ordering business process.

Figure 2–1 Overall TSO Business Process



The following paragraphs refer to the flow elements in Figure 2–1.

To begin the Telecommunications Services Ordering business process, create a quote (**1**). To view the expanded diagram, see Section 2.1.1, "Create a Quote" on page 2-3.

To add a new service for an existing customer, select an item from the Oracle Inventory Product Catalog and configure it (**2a**). To view the expanded diagram, see Section 2.1.2, "Configure New Items" on page 2-5.

To reconfigure a customer's existing service, select the item to move, change, add to, or disconnect from Oracle Install Base and reconfigure it (**2b**). To view the expanded diagram, see Section 2.1.3, "Reconfigure a Customer's Installed Services" on page 2-6.

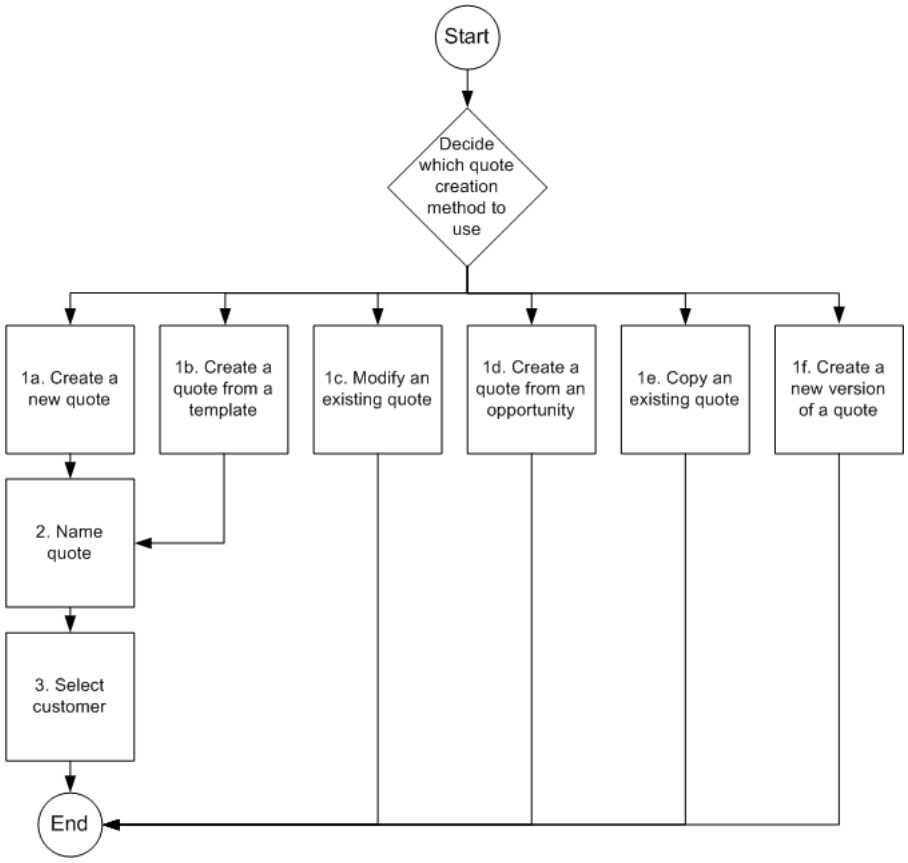
To place the sales order of the configured or reconfigured item, save the configuration and convert the quote to a sales order (**3**). After you place the sales order, the order fulfilment process begins. To view the expanded diagram, see Section 2.1.4, "Convert the Quote to a Sales Order and Verify Order Fulfillment" on page 2-8.

2.1.1 Create a Quote

When a customer requests an addition or change to a service, create a quote to begin the Telecommunications Services Ordering process.

Figure 2–2 shows the choices that are available when creating a quote.

Figure 2–2 Choices for Creating a Quote



The following paragraphs refer to the flow elements in Figure 2–2.

When you create a quote (**1a**) or create a quote from a template (**1b**), you must specify the quote name (**2**) and the customer (**3**). If the customer has multiple locations, you select a specific customer location. Many of the other fields contain default values, which you can change. Verify that the Order Type for your quote is Service Fulfillment Order before continuing.

When you modify an existing quote (**1c**), create a quote from an opportunity (**1d**), copy an existing quote (**1e**), or create a new version of a quote (**1f**), the quote name, customer, and other information originate from the source opportunity and quote, respectively.

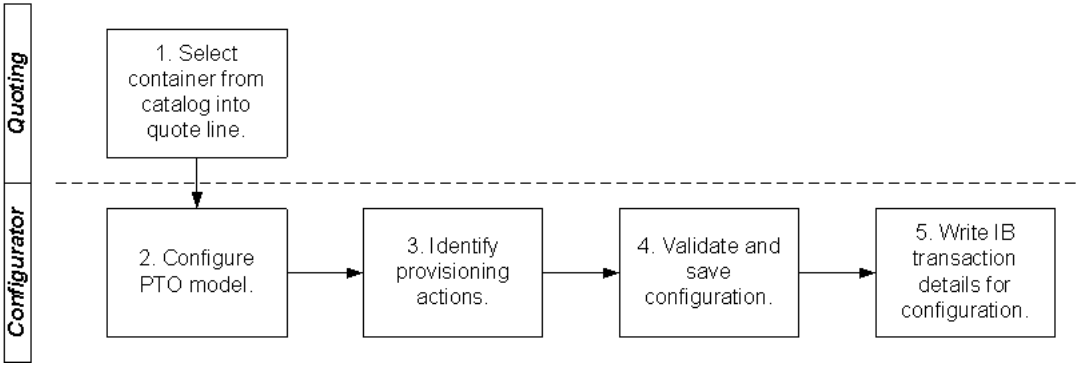
For detailed information about creating quotes using the above methods, refer to the *Oracle Quoting User Guide*.

2.1.2 Configure New Items

You begin the process of configuring a new item by selecting a configurable item from the Oracle Inventory Item Catalog and creating a quote line. From the quote line, you can access Oracle Configurator.

Figure 2–3 shows the process of configuring new items.

Figure 2–3 Process for Configuring New Items



The following paragraphs refer to the flow elements in Figure 2–3.

From your quote, you add items--products or services--to the quote from either the Oracle Inventory Item Catalog (**1**) or Oracle Install Base. To add to the quote, select a configurable Container Model from the Item Catalog. Each selected item creates a quote line.

For more information about:

- Adding items from Oracle Install Base, see Section 2.1.3, "Reconfigure a Customer's Installed Services" on page 2-6.
- Container Models, see Section 6.5, "Create a Container Model" on page 6-6.
- Creating quotes, see Section 14.2, "Create a New Quote" on page 14-2.

During the Oracle Configurator session, you can configure Pick to Order model items by selecting their attributes, locations, relationships between items, and by specifying the instance names for the selected items (**2**). Configurator identifies for

every item in the configuration a provisioning action, for example, add or disconnect (3). Oracle Configurator validates the configuration against defined technical and business configuration rules. When you are satisfied with the configuration, you submit your configuration (**4**), which results in:

- The Oracle Configurator window closing and:
  - Passing the changes back to Oracle Quoting.
  - Passing information about the transaction lines associated with each item to Oracle Install Base (**5**).
- Multiple quote lines, one for each item within the context of the configuration appearing in the quote.

You can expand or collapse the quote lines of a configuration in Oracle Quoting by clicking an expand/collapse icon on the root level of the quote line. The quote lines can also display the actions derived and the instance name captured for each item during the configuration session.

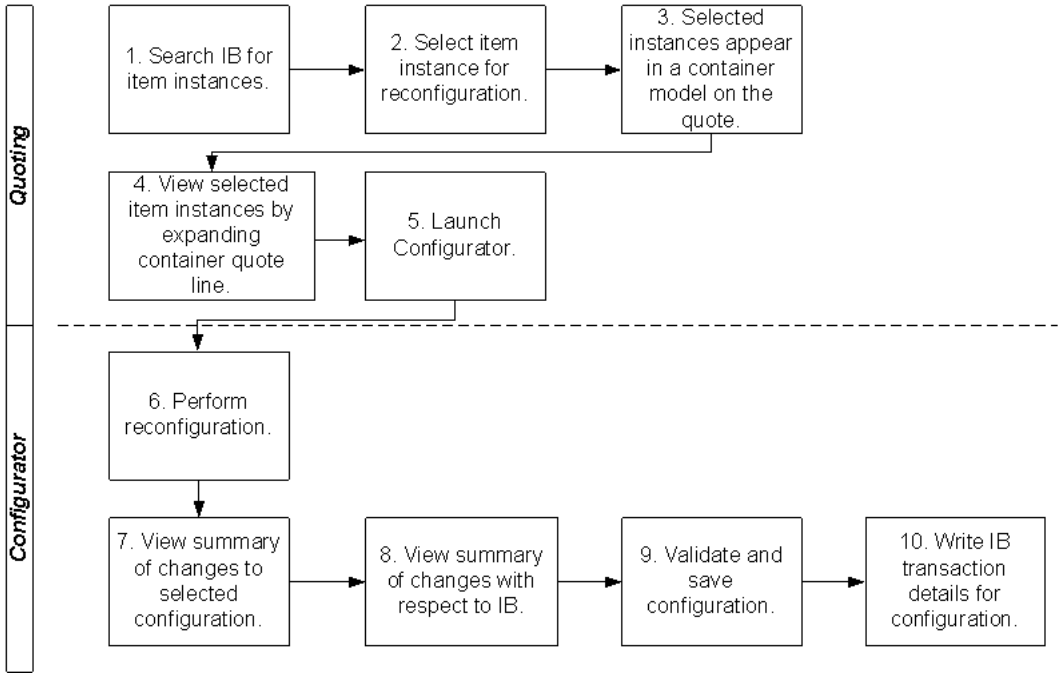
2.1.3 Reconfigure a Customer's Installed Services

You begin the process of reconfiguring item instances by selecting a configurable item instance from Oracle Install Base. Item instances created in a common Container Model appear in a single quote line. You access Oracle Configurator from the quote line.

When reconfiguring an item, consider whether you want to charge the customer for the reconfiguration. If you make changes to an installed item in the field, you can update the configuration directly from Oracle Install Base without involving Order Management and Service Fulfillment Manager.

Figure 2–4 shows the process of reconfiguring items.

Figure 2–4 Process of Reconfiguring Items



The following paragraphs refer to the flow elements in Figure 2–4.

From your quote, add items to the quote. You can select a product from either Oracle Install Base (**1**) or the Oracle Inventory Item Catalog. For more information about adding items from the Inventory Item Catalog, see Section 2.1.2, "Configure New Items" on page 2-5.

When you add items from Install Base, you navigate to specific items appearing in the selected Container Model (**2**).

The item instances that you select from Oracle Install Base appear in the quote line within their original Container Model. (**3**).

Expand the quote line to view the chosen item instances, hierarchy, and components associated with the quote line (**4**).

Next, launch Oracle Configurator (**5**). Oracle Configurator retrieves the baseline configuration status from Oracle Install Base. The baseline configuration represents the latest revision of a configurable item and reflects the current installation at a customer's site.

Reconfigure items by selecting, disconnecting, or changing their attributes, locations, and relationships between items (**6**). Oracle Configurator validates the changed configuration against defined configuration rules.

Optionally, compare the reconfiguration to the previous configuration in the Configurator Summary window. The Summary window shows the changes since the previously fulfilled, baseline configuration. From the View list of the Summary page you can choose either:

- **Current Selections:** This option displays the configuration model's entire structure and highlights all items that have changed (**7**) compared to the installed baseline configuration o r in the current configuration session.
- **Changes Relative to Installed Configuration:** This option shows only the parts that have changed and displays their ancestors for context. Parts of the configuration model that have not changed do not appear in this view (**8**).

The Summary window indicates the type of change in the Line Type column. When an Oracle Quoting user submits a quote, creating a sales order in Oracle Order Management, the provisioning action for each quote and order line appears in the Line Type column in the Order Management Sales Orders window. Oracle Order Management also passes the Line Type to downstream applications--such as Oracle Service Fulfillment Manager--for further processing.

When you are satisfied with and have completed the configuration (**9**):

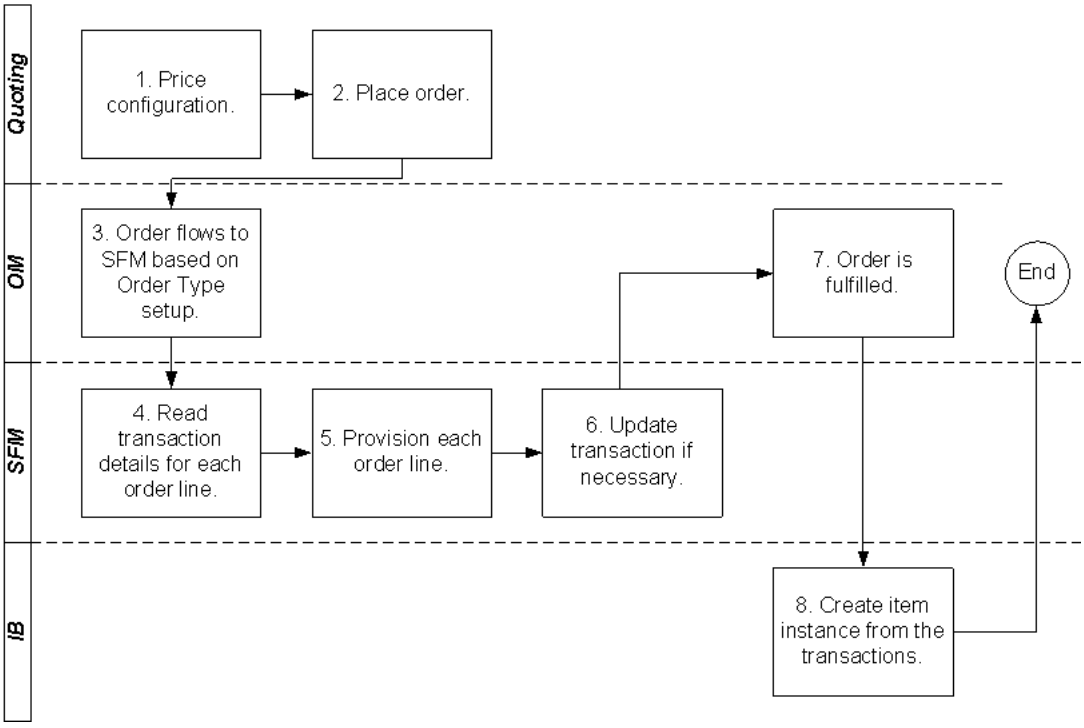
- The Oracle Configurator session ends.
- Oracle Quoting extracts the configuration information from Oracle Configurator to create quote lines, one for each item of the configuration. You can expand or collapse the quote lines of a configuration in Oracle Quoting by clicking an expand or collapse icon on the root level of the quote line.
- Oracle Configurator writes transaction details of the configuration to the Oracle Install Base (**10**).

2.1.4 Convert the Quote to a Sales Order and Verify Order Fulfillment

After you configure items, default prices appear in the quote lines. You have the option to adjust prices, along with some other quote information before booking a sales order. When you place the sales order in Quoting, Order Management receives the sales order. Oracle Order Management passes the sales order information on to Service Fulfillment Manager for provisioning. When you provision the order, Oracle Install Base receives the updated item information and the sales order is complete.

Figure 2–5 shows the process of converting a quote to a sales order and verifying order fulfillment.

Figure 2–5 Converting a Quote to a Sales Order and Verifying Order Fulfillment



The following paragraphs refer to the flow elements in Figure 2–5.

Pricing Items

When you finish configuring an item, Oracle Advanced Pricing prices the selected items; these prices appear in the quote line (**1**). You can choose which columns display on the Lines page and change the column labels to best meet your needs. For more information, see Section 12.3, "Set Up the Lines Page" on page 12-6.

Changing the Quote

You have the option of changing the following quote information at this time:

- Change the price of an item on the quote by:

- Providing a manual discount.
- Adjusting the price of a quote line by applying a promotion or clicking a price list.
- Applying a pricing agreement.

Note: If a line does not include a specified price list, then the header's price list is the default. If the header does not include a price list, then Oracle Advanced Pricing determines the price list.

- Change the shipping and billing information for a selected quote line.

Note: If a product or service item does not include a specified Installed At address during the configuration session, then the shipping address becomes the default Installed At address in Oracle Install Base.

- Specify the expiration date of the quote.

Submitting the Quote for Approval

Submit the quote for approval using Oracle Approvals Management. Define the approval Rules and Approvers during the setup process. The quote remains in read-only mode during the approval process. Oracle Approvals Management submits requests for approval to each approver in sequence, sending an e-mail notification to you upon completion of the approval process. During the approval process, you can view the response of each approver. You can also cancel an approval request that is in progress, which sends e-mail notification to the approver with whom the approval request is pending.

Placing the Order

When you are satisfied with the quote, you can:

- Place an order for the quote (**2**).
- Print the quote.
- Freeze the quote prices.
- Create a new version of the quote.

– Create a proposal from the quote.

Although you can place an order in either Booked or Entered status, the Telecommunications Service Ordering solution requires that you place the order in Booked status.

After Placing the Order

After you place an order from the quote, Oracle Order Management creates an order in Order Manager (3). Although you can place an order in either Booked or Entered status, the Telecommunications Service Ordering solution requires that you place the order in Booked status.

After placing an order in Booked status, all of the order lines with provisionable items flow into the Service Fulfillment Manager (4).

> **Note:** If you place an order from a quote in Entered status, the order lines requiring provisioning do not flow into SFM until you change the status of the order to Booked. Changing the status of an order requires you to access the order in Oracle Order Management and manually change the status from Entered to Booked. The Telecommunications Services Ordering solution requires that you set the Oracle Quoting profile option **ASO: Default Order State** to a default order status of **Booked**.

Oracle Service Fulfillment Manager fulfills each order line by performing the provisioning action--appearing in the Line Type field--on the item. (5).

After provisioning the service on the network, Oracle Service Fulfillment Manager:

- Records all of the activation information and modifications performed during the fulfillment process.
- Updates the Oracle Install Base transaction details schema if any configuration attributes that changed during provisioning, along with the fulfillment date of each order line (6). A workflow updates the fulfillment date. For more information about the workflow processes associated with the Oracle Order Management Order Line item type, refer to *Using Oracle Workflow in Oracle Order Management*.

The Oracle Install Base and Oracle Order Management integration process either:

- Creates instances in Oracle Install Base based on the fulfilled transaction details, or

- Makes updates if any configuration attributes changed during provisioning (8).

Before creating or updating an instance, however, Oracle Install Base runs the Oracle Configurator batch validation process to validate the instance.

When SFM completes provisioning, the status of the order line becomes Provisioning Successful and then Fulfilled (7).

See Also

You can find detailed end-user procedural steps in Chapter 14, "Using the Solution". Topics include:

- Section 14.2, "Create a New Quote" on page 14-2
- Section 14.3, "Add Products to the Quote" on page 14-4
- Section 14.4, "Reconfigure an Existing Service" on page 14-5
- Section 14.5, "Add Products to Quote from Install Base" on page 14-6
- Section 14.6, "Add to Container Model from Install Base" on page 14-7
- Section 14.7, "Remove Unchanged Components" on page 14-8
- Section 14.8, "Remove Lines" on page 14-9
- Section 14.9, "Place an Order" on page 14-10

Part II

Functionality and Setup

This part contains the following chapters:

- Chapter 3, "Process Functionality"
- Chapter 4, "Setup Flow and Checklist"
- Chapter 5, "Set Up Order Management"
- Chapter 6, "Set Up Inventory"
- Chapter 7, "Set Up Bill of Materials"
- Chapter 8, "Set Up Install Base"
- Chapter 9, "Set Up Advanced Pricing"
- Chapter 10, "Set Up Configurator and Customize the Solution"
- Chapter 11, "Set Up Configurator Functional Companions"
- Chapter 12, "Set Up Quoting"
- Chapter 13, "Set Up Service Fulfillment Manager"

3

Process Functionality

Main topics in this chapter include:

- Section 3.1, "Key Functionality of the TSO Solution" on page 3-3
- Section 3.2, "Quoting" on page 3-3
- Section 3.4, "Configurator" on page 3-7
- Section 3.5, "Install Base" on page 3-11
- Section 3.6, "Service Fulfillment Manager" on page 3-12

3.1 Key Functionality of the TSO Solution

The Oracle Telecommunications Service Ordering (TSO) solution spans multiple Oracle applications. Each application has developed a set of functionality that when used together enables the TSO solution. The purpose of the this document is to provide the reader with an understanding of the solution, the overall user flow, the functionality and the setup required on each Oracle Application to enable to TSO solution.

The TSO solution is available on an 11.5.9 release of Oracle Applications with the Order Management ARU 2529922.

Key functionality of the Telecommunications Service Ordering solution includes:

Order Management

The Order Management ARU 2529922 allows the components of a configuration to have different Line Types. Without this ARU, when a quote containing a configuration, where the components have different Line Types (or Actions), is booked, the Line Type of the top-most item in the configuration will overwrite the Line Type of the component lines.

Oracle Quoting

- Supporting Container Models. This involves solution-based modeling, network connections (or links), and partial configurations.
- Searching Oracle Install Base for the purpose of reconfiguring existing installed instances.
- Launching Oracle Configurator to add new instances or reconfigure installed instances to the quote.

For more functionality information, see Section 3.2, "Quoting" on page 3-3.

Oracle Configurator

- Configure new items and create new configurations.
- Restoring configurations from Oracle Install Base.
- Reconfiguring existing configurations.
- Determining the differences between the installed and the reconfigured configuration.
- Submitting the reconfigured configuration to the hosting application--Oracle Order Management, Oracle Quoting, or Oracle Install Base.

For more functionality information, see Section 3.4, "Configurator" on page 3-7.

Oracle Install Base

- Storing network configuration models.
- Identifying and maintaining the current baseline configuration.
- Launching and validating updates to the baseline configuration.
- Managing connected-to relationships.

For more functionality information, see Section 3.5, "Install Base" on page 3-11.

Oracle Service Fulfillment Manager

Oracle Service Fulfillment Manager provisions the telecommunication services that the customer ordered.

For more functionality information, see Section 3.6, "Service Fulfillment Manager" on page 3-12.

3.2 Quoting

Functionality topics in Quoting include:

- About Quoting on page 3-3
- Supporting Container Models on page 3-4.
- Search Oracle Install Base on page 3-4.
- Launch Oracle Configurator on page 3-5.

About Quoting

Oracle Telecommunications Service Ordering allows sales agents to add or reconfigure telecommunication services in the customer's installed base. When a customer wants to make changes or add components to a product configuration that they have ordered and received, the sales agent can either search Oracle Install Base and select the customer's telecommunication service or search Oracle Inventory to add new items. The sales agent can launch Oracle Configurator from Oracle Quoting and either revise the original configuration or configure the new items.

Oracle Configurator returns a Line Type or Action, to Oracle Quoting to inform the user if a change has been made. The Container Model and all components have the

default line category ORDER. After configuration, the pricing of the quote and placement of the order occur. When the order fulfillment occurs through Oracle Service Fulfillment, updates to the configuration occur in Oracle Install Base.

Supporting Container Models

Oracle Quoting supports the ordering of an entire telecommunications network within one configuration. When your customer requests a quote for network items or services, you can individually configure multiple occurrences of a model or component (solution-based modeling) and ensure the connection of all network components before finalizing the configuration (network connections). When a customer with an existing network requests changes within the network, Oracle Quoting also supports the reconfiguration of portions of the network (partial configurations).

Solution-based modeling enables the use of multiple instantiation, which is the ability to create and individually configure multiple occurrences of a model or component in a runtime Oracle Configurator session. Each individually configured item instance appears underneath the Container Model quote line when you expand the quote line.

Network connections or links enable you to specify how to connect some or all of the components of a configuration. You can establish network connections when you access Oracle Configurator through Oracle Quoting.

You can quote the partial configuration of an existing network when a customer requests a move, addition, change, or disconnection within the network. From Oracle Quoting:

- Select the current network configuration within Oracle Install Base.
- Launch Oracle Configurator to make changes to the current configuration.

For more information about solution-based models and connectivity, see the *Oracle Configurator Developer User's Guide*.

Search Oracle Install Base

Oracle Install Base stores the current configuration for customers' existing networks. When a customer requests a quote to change an existing configuration, you can search Oracle Install Base for the existing configuration from Oracle Quoting and launch Oracle Configurator.

Oracle Configurator rules control the availability of modifications to the existing configuration. After completing the reconfiguration, return to Oracle Quoting where the Action or Line Type column shows the provisioning action performed on

each telecommunication service line during reconfiguration. Oracle Quoting calls Oracle Pricing to get updated pricing for the model and components. You can remove **unchanged components** if the customer only wants to view changed items. For more information on unchanged components, see Section 14.7, "Remove Unchanged Components" on page 14-8.

Launch Oracle Configurator

Oracle Quoting can launch Oracle Configurator either directly, when you select a model for configuration from Oracle Inventory, or after retrieving an existing configuration from Oracle Install Base.

3.3 Pricing

When a configuration is validated and saved by Configurator, it must be priced. Pricing of the configuration can be done before the configuration appears on the quote (such as, automatic) or after the configuration appears on the quote (such as, batch mode).

In the first case, Advanced Pricing is called after the configuration is complete and the quote lines and the prices appear together on the quote. If the configuration is very large, or the pricing very complex, then the user sees the Saving Configuration window until pricing is complete.

When a quote is priced in batch mode, the saved configuration appears on the quote lines before Advanced Pricing is called to price the quote. The user can then price the quote. While the quote is being priced, the user can view the configuration on the quote lines. If this option is used, the user must explicitly price the quote to view the quote prices and to apply price adjustments.

Oracle Quoting provides the capability to edit the labels of seeded columns in the Lines and Overview pages. It also allows you to hide or display seeded columns, and change the order of the display of the columns on these pages. This functionality is used in the Oracle Telecommunications Services Ordering solution to display Recurring Charges and One-Time Charges.

Recurring Charges

Charges that a customer pays the telecommunications service provided on a periodic basis are recurring charges. Examples of recurring charges are $29.99 per month for a Wireless Phone Service or $5000.00 per year for a full T1 line. The period of recurrence for the Wireless Phone Service is every month, and the period of recurrence for the full T1 is once a year.

The Recurring Price of a service is captured as the Unit List Price of the item representing the service. The label of the Unit List Price column in Oracle Quoting quote lines page changes to **Recurring Charges**. The Unit Of Measure (UOM) of the item representing the telecommunications service represents the period of recurrence for the charge. To setup the unit list price of an item, refer to the *Oracle Advanced Pricing User Guide* for information on adding items to a price list

Discounts on Recurring Charges

You apply discounts to the Recurring List price of a telecommunications service by specifying the discount in the Unit Adjustment Percent column on the Oracle Quoting Lines page. You can change the label of the Unit Adjustment Percent column to **Recurring Adjustment**. For more information on customizing columns in the quote, see Section 12.3, "Set Up the Lines Page" on page 12-6.

One-Time Charges

A fixed charge that a customer pays the telecommunications service provider only once is a **one-time charge**. Some examples of one-time charges include Activation Fees, Installation Fees, and fees to switch long-distance carriers.

The one-time charge of a telecommunications service is modeled as a Freight and Special Charge modifier of the inventory item representing the service. The label of the seeded Charges column in the Oracle Quoting quote lines page is changed to read **One-time Charges**. For more information on how to change column names, see Section 12.3, "Set Up the Lines Page" on page 12-6. For more information on setting up charges, see Section 9.1, "Set Up One-Time Charges" on page 9-3.

You can define Multiple Charges (or Charge Names) for an item. To define Multiple Charges (or Charge Names) for an item, you create a pricing modifier for each Charge Name. When the charge is applied to the item, it is controlled by defining a Pricing Qualifier on the Pricing Modifier. For a detailed description of Pricing Modifiers and Qualifiers, refer to the *Oracle Advanced Pricing Implementation Guide*. The Pricing Setup section of this document also provides details on setting up modifiers and qualifiers for telecommunications items.

Discounts on One-Time Charges

You can give discounts on One-Time Charges by using the standard QP functionality **`GET_CUSTOM`** and the Order Capture descriptive flexfield, **`LINE: Additional Information`**.

Recurring Totals

The Lines page in Oracle Quoting shows:

- The sum of recurring charges.
- The recurring charges as the **Sub-Total (List Price)** in the Grand Total section.

One-Time Charge Totals

The Lines page in Oracle Quoting shows:

- The sum of the one-time charges.
- The one-time charges as a link against the Charges row of the Grand Total section. The user can view a break-down of all the one-time charges by clicking the link.

3.4 Configurator

The key topics of Configurator functionality that enable the Upgrade Process Flow for telecommunications service networks include:

- Integration with Oracle Install Base on page 3-7.
- Partial Network Reconfiguration and Validation on page 3-7.
- Computation of Configuration Deltas for Networks on page 3-9.
- Identification and Fulfillment Based on User-Defined Actions (Move, Add, Change, Disconnect) on page 3-10.
- Persistence of User-Defined Attribute Values Throughout the Fulfillment Process on page 3-10.

Integration with Oracle Install Base

With the integration with Oracle Install Base, you can track the full life cycle flow of a product as it is uniquely configured, ordered, fulfilled, serviced, and upgraded over time. Oracle Install Base serves as the central repository for storing all product information. In performing any type of product upgrade, Oracle Install Base is called upon for the most up-to-date product data. Configurator uses the Install Base data as the **baseline** when performing product upgrades or reconfiguration.

Partial Network Reconfiguration and Validation

Partial network reconfiguration and validation invokes only the component(s) that the end user chooses to upgrade. Even though only potentially those chosen components are actively restored within the runtime configuration session,

Configurator validates any proposed changes against all impacted components to ensure a valid state of the complete network.

In terms of the runtime environment, the end user can now identify in Oracle Quoting that he or she wants to upgrade existing services as opposed to purchasing new ones. A new Install Base Search window appears where the end user can perform queries based on a variety of different search criteria. The Search window returns all component(s) that match the end user's search criteria, and the end user can choose the component(s) to upgrade. Configurator launches with only those component(s) that are directly impacted--the chosen component(s) (active component)--and those components that could be indirectly impacted, also called passive components.

Figure 3–1, "Configurator's Runtime View" shows Configurator's runtime view when updating chosen Frame Relay Service components.

Figure 3–1 Configurator's Runtime View

The preceding graphic shows that the end user chose to update both the Boston location (Port) and the connection from Boston to NYC (PVC). The chosen Port and PVC are invoked in Configurator in a read-write mode so that the end user can make changes to them. Configurator is also invoked with the read-only NYC Port. Due to the fact that the PVC connects the Boston and NYC locations, Configurator also restores the NYC location to ensure that changes made in either the Boston location or the PVC connection do not invalidate any options or rules associated with the NYC location. The end user also can activate any read-only (passive)

component, such as the NYC location, in the runtime configuration session. Any changes are validated against the entire existing system.

The major benefits to this partial network reconfiguration and validation functionality are:

- The end user can update sub-components independently while still ensuring validity throughout the entire network
- Runtime performance is based on the size of the component(s) to be updated. When a company is selling complex products, such as Frame Relay Services, consisting of thousands of locations and thousands of connections, it is a major performance advantage to be able to invoke only those components that the end user chooses to update.

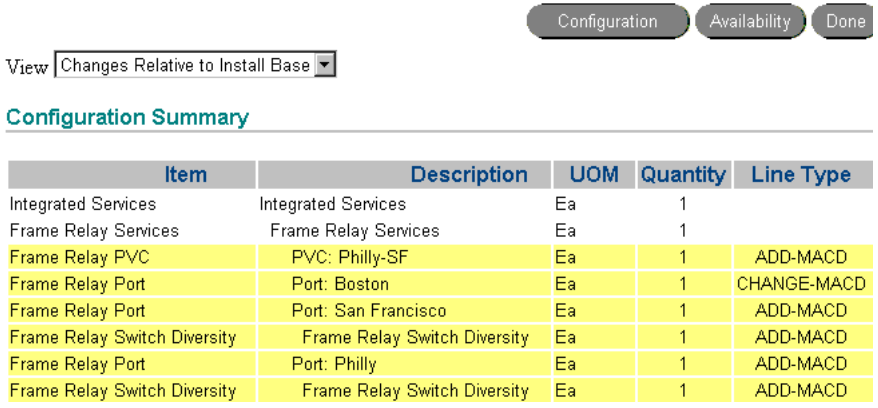
Computation of Configuration Deltas for Networks

Computation of configuration deltas for networks is the ability to identify the current state of a configuration as the **baseline**, allow the end user to upgrade from this current state, and to quote, price, and fulfill only those changes. In terms of the runtime environment, Configurator is invoked with the options and items of the selected current state. The end user can change these options and items to meet new requirements. The following graphic shows Configurator associating the changes with the appropriate Line Type or Action, and it also allows the end user to choose from various delta views. The delta views are:

- The current selections.
- Changes relative to Install Base.

Figure 3–2 shows Configurator's Summary window displaying the various delta views along with actions and Line Type associations.

Figure 3–2 Configuration Summary



Item	Description	UOM	Quantity	Line Type
Integrated Services	Integrated Services	Ea	1	
Frame Relay Services	Frame Relay Services	Ea	1	
Frame Relay PVC	PVC: Philly-SF	Ea	1	ADD-MACD
Frame Relay Port	Port: Boston	Ea	1	CHANGE-MACD
Frame Relay Port	Port: San Francisco	Ea	1	ADD-MACD
Frame Relay Switch Diversity	Frame Relay Switch Diversity	Ea	1	ADD-MACD
Frame Relay Port	Port: Philly	Ea	1	ADD-MACD
Frame Relay Switch Diversity	Frame Relay Switch Diversity	Ea	1	ADD-MACD

Identification and Fulfillment Based on User-Defined Actions (Move, Add, Change, Disconnect)

Configurator associates changes with appropriate Line Types. For example, in Figure 3–2, changing the configuration by adding a Frame Relay PVC: Philly-SF specifies ADD-MACD. You define the Line Type ADD-MACD when you implement the TSO solution. Furthermore, the TSO implementer can tie these Line Types to the appropriate fulfillment steps.

Persistence of User-Defined Attribute Values Throughout the Fulfillment Process

Persistence of user-defined attribute values throughout fulfillment process attribute values such as custom names and location to one global instance (Install Base), so that all applications can have access to the most up-to-date information. These values are stored and updated throughout the order creation to order fulfillment process to ensure timely and accurate service fulfillment.

The Oracle TSO solution supports integration with Install Base, the computation of deltas, and Line Types, the support of attributes, and the partial reconfiguration of components only when the components are children of a Container Model.

For information about defining configuration models in general, see the *Oracle Configurator Developer User's Guide*.

3.5 Install Base

When you complete ordering and fulfillment of the Container Model defined in Oracle Inventory, Oracle Bills of Material, and Oracle Configurator, the saved configuration appears in Oracle Install Base as item instances. You can view or reconfigure the configured item instances, their attributes, and relationships from either Oracle Install Base or Oracle Quoting, which accesses Oracle Install Base.

Refer to the *Oracle Install Base Implementation Guide* and *Oracle Install Base Concepts and Procedures* for more information about Oracle Install Base.

Functionality topics in Install Base include:

- Storing Network Configuration Models on page 3-11
- Identifying and Maintaining the Current Baseline Configuration on page 3-11
- Managing Connected-to Relationships on page 3-12

Storing Network Configuration Models

Oracle Install Base can support a network connection at multiple service locations. This includes networks implemented in rings; for example, a network where A is connected to B, B is connected to C, and C is connected to A.

Identifying and Maintaining the Current Baseline Configuration

Oracle Configurator validates every new configured service or changed service before the service is saved in Oracle Install Base. You can initiate service configuration changes from a quote (most common) or from Oracle Install Base. Changes can include moves, additions, changes, or disconnections to an existing service. Oracle Quoting retrieves the existing configuration from Oracle Install Base. You can launch Oracle Configurator from either Oracle Quoting or Oracle Install Base and reconfigure the service.

Before saving the configuration to Oracle Install Base, processing of the revised configuration includes:

- Reconfiguring a customer's installed services. For more information, see Section 2.1.3, "Reconfigure a Customer's Installed Services" on page 2-6.
- Converting the quote to a sales order. For more information, see Section 2.1.4, "Convert the Quote to a Sales Order and Verify Order Fulfillment" on page 2-8.
- Verifying order fulfillment. For more information, see Section 2.1.4, "Convert the Quote to a Sales Order and Verify Order Fulfillment" on page 2-8.

Managing Connected-to Relationships

Oracle Configurator uses connected-to relationships to make network connections and show the service configuration. You can view the connected-to relationships of a configured instance from Oracle Install Base. If you define items in Oracle Inventory as link items, you can use Oracle Install Base to show the start and end locations of the link. The locations for the link instance are the geographic addresses of the instance items. For information about defining an inventory item as a network link, see Section 6.3, "Set an Item as Provisionable" on page 6-4.

3.6 Service Fulfillment Manager

Oracle Service Fulfillment Manager provisions the telecommunication services that the customer ordered by:

1. Capturing a service order request.
2. Validating the order.
3. Analyzing the order.
4. Fulfilling the order.
5. Completing the order.

And, if necessary,

6. Managing order fallout.

Refer to *Oracle Service Fulfillment Manager Concepts and Procedures* for a description of the provisioning process. The provisioning functions are the same for the Oracle Telecommunications Service Ordering solution as they are for any other service order request that Oracle Service Fulfillment Manager receives. However, Oracle Service Fulfillment Manager requires some specific setup steps to receive a configured order from Oracle Order Management and to retrieve and update extended attributes from the Oracle Install Base Transaction Details window.

For more information about setting up Oracle Service Fulfillment Manager, see Chapter 13, "Set Up Service Fulfillment Manager".

4

Setup Flow and Checklist

Topics in this chapter include:

- Section 4.1, "About Setting Up Telecommunications Service Ordering" on page 4-2
- Section 4.2, "Setup Flow" on page 4-2
- Section 4.3, "Setup Checklist" on page 4-4

4.1 About Setting Up Telecommunications Service Ordering

This document includes setup information that is **specific** to the Oracle Telecommunications Service Ordering (TSO) solution.

The Oracle Telecommunications Service Ordering functionality requires the use of these Oracle Applications:

- Oracle Order Management
- Oracle Inventory
- Oracle Bill of Materials (BOM)
- Oracle Install Base
- Oracle Configurator
- Oracle Quoting
- Oracle Service Fulfillment Manager (SFM)

For complete information on setting up the preceding applications, refer to that Oracle Application's documentation.

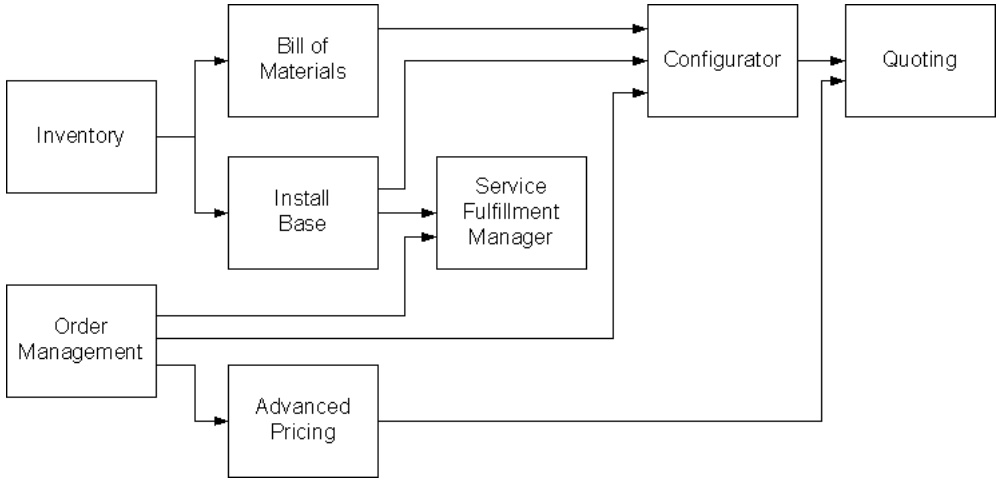
4.2 Setup Flow

The setup flow shows the Oracle Application sequence and their dependencies for setting up the Oracle Telecommunications Service Ordering solution.

Note: This document provides additional setup information that is not included in the implementation guides of the Oracle Applications that are required to support this release of Oracle Telecommunications Service Ordering. The setup and design tips apply primarily to Oracle Configurator, Oracle Quoting, Install Base, and Service Fulfillment Manager. You should use this document in conjunction with the implementation guides of each of these Oracle Application.

Figure 4–1 shows the TSO setup flow by Oracle Application product.

Figure 4–1 TSO Setup Flow by Oracle Application



For example,

- First, set up both Order Management and Inventory.
- After you set up Inventory, you can set up both Bill of Materials and Install Base.
- After you set up Order Management, Bill of Materials, and Install Base, you can set up Configurator.
- After you have set up both Order Management and Install Base, you can set up Service Fulfillment Manager.
- After you have set up Order Management, you can set up Advanced Pricing.
- After you have set up both Configurator and Advanced Pricing, you can set up Quoting.

4.3 Setup Checklist

The following is a setup checklist. Note that you can set up some applications in different sequences, since the various Oracle Applications that are part of the Oracle Telecommunications Service Ordering solution have different dependencies and setup flows. For more information, see Section 4.2, "Setup Flow" on page 4-2.

Table 4–1 TSO Setup Check List

Chapter	Setup
Set Up Order Management	-
	Ensure Application of Patch 2529922 on page 5-3
	Modification to OM Workflow Processes on page 5-3
	Create SFM Enabled Order Header and Line Workflows on page 5-3
	Replace the Fulfill Node in the OM Line Workflow on page 5-3
	Set Up Transaction Types on page 5-4
	Set Up Line Transaction Type on page 5-4
	Set Up Order Transaction Type on page 5-5
	Design Tips on page 5-6
Set Up Inventory	-
	Setup of Telecommunications Service Items on page 6-3
	Specify Install Base Tracking for Each Component Item on page 6-3
	Set an Item as Provisionable on page 6-4
	Set Up an Item of Type Link on page 6-5
	Create a Container Model on page 6-6
	Specify Container Model on page 6-6
	Specify Install Base Tracking for Each Component Item on page 6-3
Set Up Bill of Materials	-
	About Setting Up BOM on page 7-2
Set Up Install Base	-

Table 4–1 TSO Setup Check List (Cont.)

Chapter	Setup
	Enable Network Configurations on page 8-3
	Create Extended Attributes on page 8-3
	Set Up Extended Attribute Pools on page 8-4
	Map Extended Attributes to Items on page 8-6
	Design Tips on page 8-7
Set Up Advanced Pricing	-
	Set Up One-Time Charges on page 9-3
	Create One-Time Charge Names on page 9-3
	Define Modifier for an Item or Service on page 9-4
	Define Qualifiers on the Modifier on page 9-7
	Discount to One-Time Charges on page 9-8
	Pricing Attributes and Sourcing Rules on page 9-9
	Get Custom Price on page 9-13
Set Up Configurator and Customize the Solution	-
	Create the Configuration Model on page 10-16
	Publish a Container Model on page 10-18
	Define Profile Options on page 10-18
	Disable Pricing on page 10-19
	CZ Schema Customizations on page 10-19
	TSO Solution CZ_DB_SETTINGS Parameters on page 10-20
	CZ_CONFIG_ATTRIBUTES Table on page 10-21
	CZ_CONFIG_EXT_ATTRIBUTES Table on page 10-21
	MACD_FULFILLMENT_STRUCTURES Table on page 10-23
	Mapping of CZ tables to Install Base Schema (CSI) on page 10-25
	Initialization Parameters on page 10-27

Table 4–1 TSO Setup Check List (Cont.)

Chapter	Setup
	Batch Validation Parameters on page 10-29
	Programmatic Tools for TSO Development on page 10-30
Set Up Configurator Functional Companions	-
	Modifying the TSO Functional Companions on page 11-3
	Overview on page 11-3
	Specify the Default Location on page 11-5
	Setup for the Default Location Functional Companion on page 11-6
	Modify the Default Location Functional Companion on page 11-7
	Specify the Line Type on page 11-9
	Setup for the Line Type Functional Companion on page 11-10
	Modify the Line Type Functional Companion on page 11-18
	Change the Instance Name on page 11-18
	Setup for the Instance Name Functional Companion on page 11-18
	Modify the Instance Name Functional Companion on page 11-21
	Using Configuration Attributes with Install Base (IB) on page 11-21
	Setup for the IBAttribute Functional Companion on page 11-22
	Modify the IBAttribute Functional Companion on page 11-23
	Using the Oracle Configuration Interface Object on page 11-25
	Update Installed Configurations on page 11-25
	Identify Changes to a Configuration on page 11-25

Table 4–1 TSO Setup Check List (Cont.)

Chapter	Setup
	Accessing Discontinued Items on page 11-27
	Accessing Instances on page 11-27
	Handling Interface Events on page 11-28
	Automatic Behavior for Configurations on page 11-31
Set Up Quoting	-
	Summary of Reconfiguration on page 12-2
	Define Profile Options on page 12-3
	Set Up the Lines Page on page 12-3
	Display or Hide Columns on page 12-8
	Hide a Seeded Column on page 12-10
	Edit Column Labels on page 12-11
	Edit Column Sequence on page 12-12
	Enable or Disable Line-Level Actions or Table-Level Buttons on page 12-13
	Design Tips on page 12-16
Set Up Service Fulfillment Manager	-
	Section 13.1, "Create SFM Enabled Order Management Workflow Header Process" on page 13-3
	Section 13.2, "Create SFM Enabled Order Management Workflow Line Process" on page 13-4
	Section 13.3, "Add Function for Menu Notifications" on page 13-5
	Section 13.4, "Create a PL/SQL Procedure That Updates a Certain Workitem Parameter" on page 13-6
	Section 13.5, "Create Workflow Process" on page 13-7
	Section 13.6, "Define Work Item" on page 13-9
	Section 13.7, "Map Work Item to Item and Action (Line Transaction Type) Combination" on page 13-10
	Section 13.8, "Design Tips" on page 13-11

5

Set Up Order Management

The main topics in this chapter are:

- Section 5.1, "Ensure Application of Patch 2529922" on page 5-3
- Section 5.2, "Modification to OM Workflow Processes" on page 5-3
- Section 5.3, "Set Up Transaction Types" on page 5-4
- Section 5.4, "Design Tips" on page 5-6

Reference Material

This chapter describes setup tasks that you must perform in Oracle Order Management (OM) that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on setting up and using Oracle Order Management, refer to:

- *Oracle Order Management Suite Implementation Manual*
- *Oracle Order Management User's Guide*

Before You Begin

Among the Oracle Applications that are directly a part of the Oracle Telecommunications Service Ordering solution, there are no dependencies to setting up Oracle Order Management. It is recommended that you set up Oracle Order Management first.

After You Set Up Order Management

After you set up Oracle Order Management, you can set up Oracle Advanced Pricing or Oracle Service Fulfillment Manager. For more information, see:

- Chapter 9, "Set Up Advanced Pricing"

- Chapter 13, "Set Up Service Fulfillment Manager"

5.1 Ensure Application of Patch 2529922

If you are a Telecommunications Service Ordering customer planning to use Oracle Configurator, Oracle Quoting, and Oracle Install Base to support updates to installed telecommunication service configurations, you must apply a stand-alone Order Management ARU (patch).

Applying this patch enables Oracle Order Management to support multiple Line Types in a configuration. The ARU number is 2529922. This ARU is under controlled release, so you must request it by contacting Oracle Support.

5.2 Modification to OM Workflow Processes

The modification to OM Workflow processes involves these topics:

- Section 5.2.1, "Create SFM Enabled Order Header and Line Workflows" on page 5-3
- Section 5.2.2, "Replace the Fulfill Node in the OM Line Workflow" on page 5-3

5.2.1 Create SFM Enabled Order Header and Line Workflows

To enable the flow of order from OM to Service Fulfillment Manager (SFM), you must make some required customizations to Order Management's Workflow processes. This enables fulfillment of provisionable items from Order Management by way of SFM.

For more information, see:

- Section 13.1, "Create SFM Enabled Order Management Workflow Header Process" on page 13-3
- Section 13.2, "Create SFM Enabled Order Management Workflow Line Process" on page 13-4

5.2.2 Replace the Fulfill Node in the OM Line Workflow

Currently the Fulfill Node:

- Performs a gating activity for components of a configuration.
- Ensures that all the lines in a configuration are fulfilled together.

As a consequence of the gating activity, the update of Install Base occurs only after fulfillment of all the components of a configuration. In the business-to-business

communications world, especially in the data services like Frame Relay, provisioning of components may occur days or months apart from each other.

Because of the gating activity of the Fulfill node in the OM Line Workflow, the updating of Installed Base does not occur until provisioning of all the components of the configuration occur. For example, this might be three months after provisioning the first component. This can result in the Install Base not accurately reflecting all the customer's current services. To avoid this issue, it is recommended that you replace the Fulfill node with one that only stamps the fulfillment date for each order line.

Important: Oracle Install Base will not process an order line without a fulfillment date. It is essential that if the Fulfillment node is removed, you **must** replace it with a node that stamps fulfilled flag, fulfillment date, and fulfilled quantity of each order line.

For more information, see:

- *Oracle Order Management Suite Implementation Manual*
- *Oracle Order Management User's Guide*

5.3 Set Up Transaction Types

In Order Management, you need to:

- Section 5.3.1, "Set Up Line Transaction Type" on page 5-4
- Section 5.3.2, "Set Up Order Transaction Type" on page 5-5

The following two procedures are examples of setting up both line and order transaction types.

5.3.1 Set Up Line Transaction Type

You must define all provisioning actions as Line Types. Provisioning actions are those actions on which you want to perform on items. Examples of provisioning actions include ADD, CHANGE, MOVE, DISCONNECT, SUSPEND SERVICE, and NO CHANGE. One of these lines types is the default action. Both Configurator and Service Fulfillment Manager use these Line Types.

Module
Oracle Applications (Forms)

Responsibility
Order Management Super Menu

Navigation
Setup > Transaction Types > Define

Steps
1. Create the Transaction Types: **MOVE**, **ADD**, **CHANGE**, and **DISCONNECT**.
2. Set Transaction Type Code to **Line**.
3. Set the Category to **Order**.
4. Set the Operating Unit to the name of your company's operations.
5. Save your work.

5.3.2 Set Up Order Transaction Type

Module
Oracle Applications (Forms)

Responsibility
Order Management Super Menu

Navigation
Setup > Transaction Types > Define

Steps
1. Create a Transaction type: **Service Fulfillment Order**.
2. Specify the Transaction Type Code.
3. Specify Order Category.
4. Specify Default Order Line Type to **ADD** or the Line Type to use as the default.
5. Specify the Operating Unit to the name of your company's operations.

6. Map the Transaction Type to Order Workflow: Order Flow - Service Fulfillment.

   For more information, see Section 13.1, "Create SFM Enabled Order Management Workflow Header Process" on page 13-3.

7. Click Assign Line Flows.

8. Assign the Line Types--such as MOVE, ADD, CHANGE, and DISCONNECT--to the Order Type **Service Fulfillment Order**.

9. Map all Line Transaction Types that you created in the previous section--such as MOVE, ADD, CHANGE, and DISCONNECT--to the Workflow Process Line Flow- Service Fulfillment.

   You created the Line Flow - Service Fulfillment process in the procedure Section 13.2, "Create SFM Enabled Order Management Workflow Line Process" on page 13-4.

10. Save your work.

11. Navigate to **Setup > Documents > Assign**.

12. Create a new Sequence Assignment for the **Oracle Order Management** application.

13. Set the Category to **Service Fulfillment Order**.

14. Specify the Set of Books.

15. Set the Sequence, such as to **Standard**.

5.4 Design Tips

You must map all actions and Line Types associated with the Configurator Line Type Functional Companion to the Order Line Type in Order Management transaction types setup.

Note: Order Management does not support reconfiguration of Container Models in an order before booking occurs. If you launch Configurator within Order Management after you have placed an order from Quoting but you have not booked it, any new lines that you create will not inherit Configurator actions as the Line Types.

6

Set Up Inventory

The main topics in this chapter are:

- Section 6.1, "Setup of Telecommunications Service Items" on page 6-3
- Section 6.2, "Specify Install Base Tracking for Each Component Item" on page 6-3
- Section 6.3, "Set an Item as Provisionable" on page 6-4
- Section 6.4, "Set Up an Item of Type Link" on page 6-5
- Section 6.5, "Create a Container Model" on page 6-6

Reference Material

This chapter describes setup tasks specific to the Oracle Telecommunications Service Ordering (TSO) solution that you must perform in Oracle Inventory to implement the TSO solution. For more information on setting up Oracle Inventory, refer to:

- *Oracle Inventory User's Guide*

After You Set Up Inventory

Except for Oracle Order Management and Oracle Advanced Pricing, most other Oracle Applications that are part of the Oracle Telecommunications Service Ordering solution have either direct or indirect dependencies on Oracle Inventory.

For more information on setting up the TSO solution in these dependent Oracle Applications see:

- Chapter 7, "Set Up Bill of Materials"
- Chapter 8, "Set Up Install Base"

- Chapter 10, "Set Up Configurator and Customize the Solution"
- Chapter 11, "Set Up Configurator Functional Companions"
- Chapter 12, "Set Up Quoting"
- Chapter 13, "Set Up Service Fulfillment Manager"

6.1 Setup of Telecommunications Service Items

When you define telecommunication service items in Oracle Inventory, you can flag them as:

- **Install Base trackable**, if you must track the life-cycle of the service in Oracle Install Base. For more information, see Section 6.2, "Specify Install Base Tracking for Each Component Item" on page 6-3.
- **A network link**, required only if you use the selected inventory item as a network link. For more information, see Section 6.3, "Set an Item as Provisionable" on page 6-4.
- **A provisionable item**, if you are going to provision or electronically fulfill the item. For more information, see Section 6.4, "Set Up an Item of Type Link" on page 6-5.
- **A Container Model**, a model that you use to contain telecommunications service products that you can reconfigure. For more information, see Section 6.5, "Create a Container Model" on page 6-6.

6.2 Specify Install Base Tracking for Each Component Item

After you have specified that the item representing your top level BOM Model is a Container Model (Section 6.5.1, "Specify Container Model"), you need to indicate whether or not you want to track each Oracle Inventory Item that is a component of the container BOM Model. Do this for each BOM Model, BOM Option Class, and BOM Standard Item to track in Oracle Install Base.

Note: The item representing your top-level BOM Model item should **not** have the selected Install Base Tracking option. Only items to track in Install Base should have the selected Install Base Tracking option.

Module

Oracle Applications (Forms)

Responsibility

Oracle Inventory

Prerequisites

Section 6.5.1, "Specify Container Model" on page 6-6

Navigation

Master Item window

Steps

1. Query the Item or define it if it does not exist.
2. Click the Service tab.
3. On the Service tab, select both the **Provisionable** and the **Installed Base Tracking** check boxes.
4. If you use the item only to link installed components and you do not usually associate the item with a location, set the **Instance Class** to **Link**.

   For example, a Permanent Virtual Circuit (PVC) does not have a specific location, but the PVC connects two Ports in a network configuration model. Upon installation of the item, Oracle Install Base derives the location from connected the components.
5. Verify that the Container Model's structure does not violate any of the requirements.

   For more information, see Section 10.1.3, "Container Model Settings and Structure" on page 10-4.
6. Click Save.

6.3 Set an Item as Provisionable

You need to specify provisionable items to ensure that all trackable items specify the **Provisionable** option. Service Fulfillment Manager--not Order Management--fulfills items that are provisionable.

Module

Oracle Applications (Forms)

Responsibility

Oracle Inventory

Steps

1. In Oracle Inventory, query the Item, or define it if it does not yet exist.
2. In the Master Item window, click the Service tab.
3. In the lower left of the tab, select the **Provisionable** check box.
4. Click Save.

6.4 Set Up an Item of Type Link

You need to specify items as the Type **Link** only if you use the selected inventory item as a network link. You must set up a configured link item with the **Install Base Instance class = Link**.

In Oracle Configurator Developer (OCD), a Model that is flagged as an item of type Link must be given exactly two Connectors.

Example

Suppose ITEM_C is an item with the IB_ITEM_INSTANCE_CLASS set to Link. If ITEM_C is connected to ITEM_A in location A and ITEM_B in location B, then a search for ITEM_C results in its connected locations, such as locations A and B, appearing as the starting and ending locations.

Module

Oracle Applications (Forms)

Responsibility

Oracle Inventory

Steps

1. In Oracle Inventory, query the Item, or define it if it does not yet exist.
2. In the Master Item window, click the Service tab.
3. In the Install Base section, choose **Link** from the Instance Class list.

> **Note:** You can select the Instance Class only if you have enabled Install Base Tracking for this item. For more information, see Section 6.2, "Specify Install Base Tracking for Each Component Item" on page 6-3.

4. Click Save.

6.5 Create a Container Model

To create a Container Model, create a model that supports updates to its installed configurable components using the following procedures:

- Section 6.5.1, "Specify Container Model" on page 6-6 to indicate that the item representing your top level BOM Model is a Container Model.
- Section 6.2, "Specify Install Base Tracking for Each Component Item" on page 6-3 to indicate whether or not you want to track each Oracle Inventory item that is a component of the Container BOM Model.

6.5.1 Specify Container Model

You need to specify that the item representing your top level BOM Model, such as the item representing the network, is a Container Model.

Module

Oracle Applications (Forms)

Responsibility

Oracle Inventory

Prerequisites

Review "Container Model Settings and Structure" on page 10-4 very carefully before defining a Container Model in Oracle Applications.

Navigation

Master Item window

Steps

1. In the Master Item window, query the Item or define it, if it does not already exist.
2. Click the Bills of Material tab.
3. On the Bills of Material tab, set the Configuration Model Type to **Container**.

Note: The default Configuration Model Type is Standard; this value means only that the item is not a Container Model.

4. Click the Service tab.
5. On the Service tab, verify that the **Installed Base Tracking** check box is ***not*** selected.

Note: If a Container Model is marked as trackable, an error occurs when you generate the **Active Model** in Configurator Developer.

6. Click the Order Management tab.
7. On the Order Management tab, select the **Pick Components** check box.

   This indicates that the Container Model is a PTO (Pick To Order) BOM Model. If a Container Model is not a PTO Model, Configurator Developer displays an error when you import it into the Configurator schema.
8. Verify that both the **Ship Model Complete** and **Shippable** check boxes are ***not*** selected.
9. Specify whether or not you want to track each Oracle Inventory Item that is a component of the Container BOM Model. For more information, see: Section 6.2, "Specify Install Base Tracking for Each Component Item" on page 6-3
10. Click Save.

7

Set Up Bill of Materials

The main topics in this chapter are:

- Section 7.1, "About Setting Up BOM" on page 7-2

Reference Material

This chapter describes setup tasks specific to the Oracle Telecommunications Service Ordering (TSO) solution that you must perform in Oracle Bill of Materials to implement the TSO solution. For more information on setting up Oracle Bill of Materials, refer to:

- *Oracle Bills of Material User's Guide*

Before You Begin

Before you can set up Oracle Bill of Materials, you must set up Oracle Inventory. For more information, see:

- Chapter 6, "Set Up Inventory".

After You Set Up Bill of Materials

After you set up Oracle Bill of Materials for use with the Oracle Telecommunications Service Ordering solution, you can set up Oracle Configurator. For more information, see:

- Chapter 10, "Set Up Configurator and Customize the Solution"
- Chapter 11, "Set Up Configurator Functional Companions"

7.1 About Setting Up BOM

In the Oracle Telecommunications Service Ordering solution, you use Bills of Material to define the structure of the Container Model and its contents. A Container Model must be a pick-to-order (PTO) model, and it can include standard items, other PTO models, and Option Classes. The Container Model and its contents must meet certain requirements. For a listing of those requirements, see "Container Model Settings and Structure" on page 10-4.

8

Set Up Install Base

The main topics in this chapter are:

- Section 8.1, "Enable Network Configurations" on page 8-3
- Section 8.2, "Create Extended Attributes" on page 8-3
- Section 8.3, "Map Extended Attributes to Items" on page 8-6
- Section 8.4, "Design Tips" on page 8-7

Reference Material

This chapter describes setup tasks that you must perform to Oracle Install Base that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on functionality, setting up, or using Oracle Install Base, refer to:

- Section 3.5, "Install Base" on page 3-11
- *Oracle Install Base Implementation Guide*
- *Oracle Install Base Concepts and Procedures*

Before You Begin

Before you can set up Oracle Install Base for use with the Oracle Telecommunications Service Ordering solution, you must set up Oracle Inventory. For more information, see:

- Chapter 6, "Set Up Inventory".

After You Set Up Install Base

After you set up Oracle Install Base, you can set up Oracle Configurator and Oracle Service Fulfillment Manager. For more information, see:

- Chapter 10, "Set Up Configurator and Customize the Solution"
- Chapter 11, "Set Up Configurator Functional Companions"
- Chapter 13, "Set Up Service Fulfillment Manager"

8.1 Enable Network Configurations

Required

You must ensure that you have set the profile option **CSI: Configurator Enabled** to **Network Models Only**. This setting allows the transfer of the Network configurations to Install Base.

Module

System Administrator

Responsibility

System Administrator

Steps

1. Navigate to Applications > System Administrator.
2. In the Navigator - System Administration window, navigate to Menu Profiles > System, and click Open.
3. In the Profile field, search for the profile **CSI: Configurator Enabled**.

   The System Profile Values window opens.
4. On the **CSI: Configurator Enabled** row in the Site column, choose **Network Models Only**.
5. Click Save.

8.2 Create Extended Attributes

Oracle Install Base supports user-definable extended attributes by item, item category, instance, and enterprise.

- **Global**: Global Level Extended Attributes are applicable to all the item instances in Install Base.
- **Item Category**: Category-level extended attributes are applicable to the items of the category for which there are defined extended attributes.
- **Inventory Item**: Item level extended attributes are applicable to all the instances of the item type for which there is a defined extended attribute.

- **Instance**: Instance level extended attributes are applicable only to the instance for which there is a defined extended attribute.

The extended attribute is created as a CSI lookup of type CSI_EXTEND_ATTRIB_POOL. When the lookup is created, it is associated with an item, item category, or instance using the Install Base Extended Attribute Template.

After the attributes are associated, they can be mapped to Configurator attributes or Service Fulfillment Manager attributes. The item attributes (features) defined in Configurator Developer are mapped to these Install Base attributes.

See Also

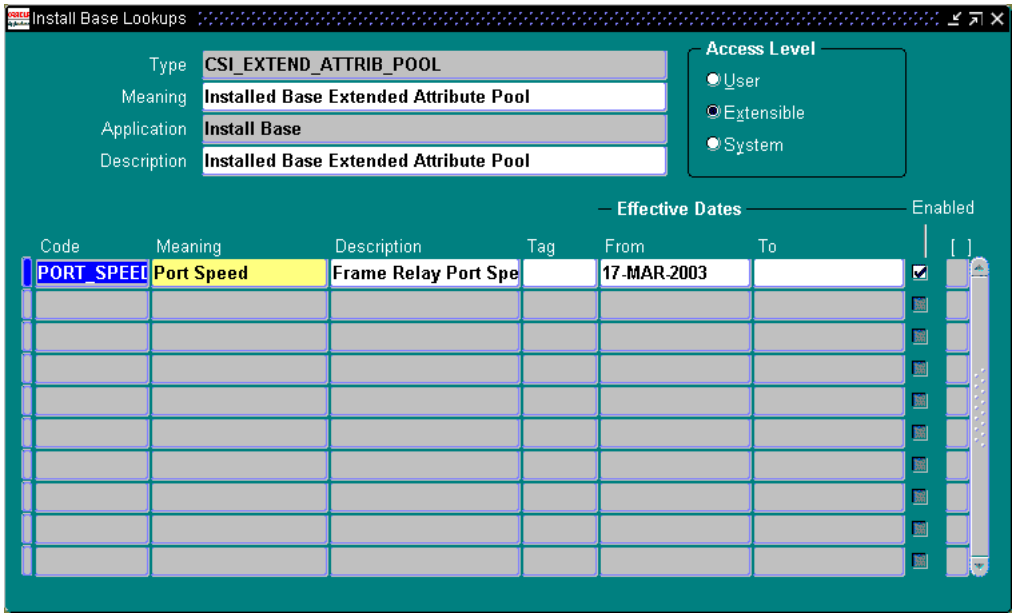
For more information on extended attribute setup, see the *Oracle Install Base Implementation Guide*.

8.2.1 Set Up Extended Attribute Pools

In the Extended Attributes window, you must define the extended attributes that the Install Base item instances use.

Figure 8–1 shows an Install Base Lookup window.

Figure 8–1 Install Base Lookup



Creation of the extended attribute is a lookup of type CSI_EXTEND_ATTRIB_POOL.

Module

Oracle Applications (Forms)

Responsibility

Oracle Installed Base Admin

Navigation

Install Base Lookups

Steps

1. Search Install Base Lookups where Type = CSI_EXTEND_ATTRIB_POOL

   A list of available lookups appears.

2. Enter a new Lookup Code, Meaning, and Description.

   Example:

   – Code = PHONE_NUM
   – Meaning = Phone Number
   – Description = Phone Number

3. Save the new lookup.

8.3 Map Extended Attributes to Items

Module
Oracle Applications (Forms)

Responsibility
Oracle Installed Base Admin

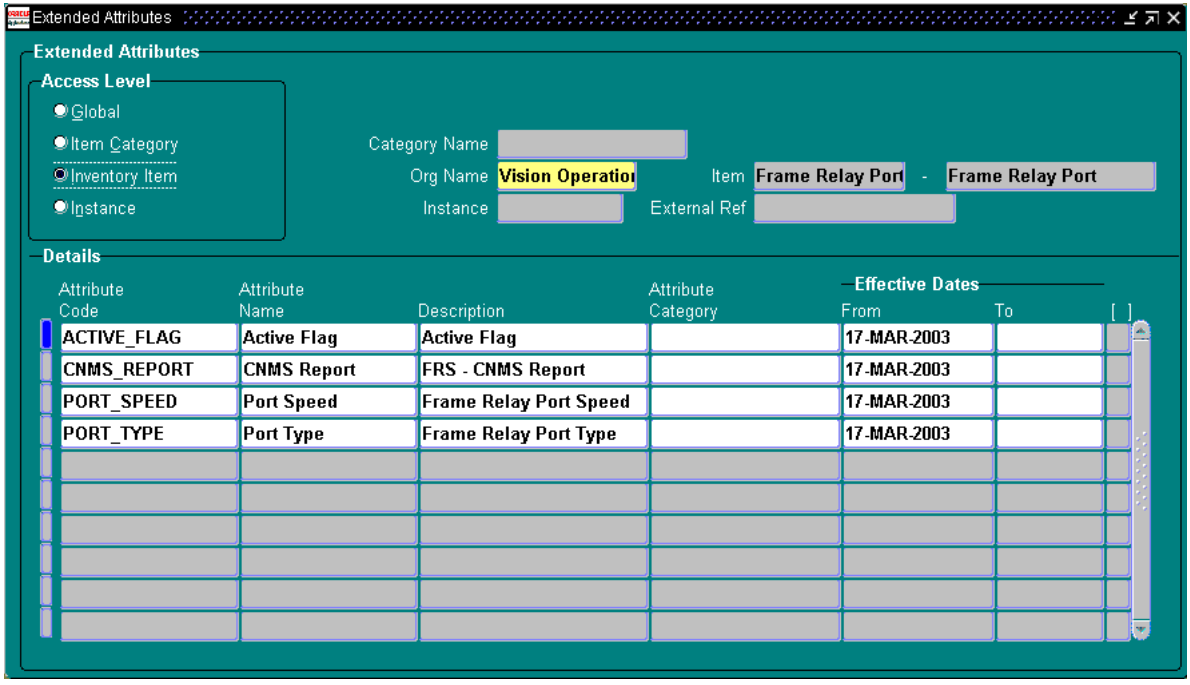
Navigation
Setups > Extended Attribute Template

Prerequisites
You must have defined the item and extended attributes.

Step
Map Extended Attributes (See Section 8.2, "Create Extended Attributes" on page 8-3) or any other attribute (Lookup) available in the CSI_EXTEND_ATTRIB_POOL to the Service Fulfillment Items.

Figure 8–2 shows the extended attribute and its details of an inventory item.

Figure 8–2 Extended Attribute And Details of an Inventory Item



8.4 Design Tips

Users of the Oracle Telecommunications Service Ordering solution must manually verify that the attributes set up in Configurator Developer are coordinated with Install Base's extended attributes setup for trackable items.

See Also

Section 10.6.3, "CZ_CONFIG_EXT_ATTRIBUTES Table" on page 10-21

Section 11.1.5, "Using Configuration Attributes with Install Base (IB)" on page 11-21

9

Set Up Advanced Pricing

The main topics in this chapter include:

- Section 9.1, "Set Up One-Time Charges" on page 9-3
- Section 9.2, "Discount to One-Time Charges" on page 9-8
- Section 9.3, "Pricing Attributes and Sourcing Rules" on page 9-9
- Section 9.4, "Get Custom Price" on page 9-13

Reference Material

This chapter describes setup tasks that you must perform to Oracle Advanced Pricing that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on functionality, setting up, or using Oracle Advanced Pricing, refer to:

- *Oracle Advanced Pricing Implementation Manual*
- *Oracle Advanced Pricing User's Guide*

Before You Begin

Before you can set up Oracle Advanced Pricing for use with the Oracle Telecommunications Service Ordering solution, you must set up Oracle Order Management. For more information, see:

- Chapter 5, "Set Up Order Management"

After You Set Up Advanced Pricing

After you set up Oracle Advanced Pricing, you can set up Oracle Quoting if have already set up Configurator. (Oracle Quoting is dependent on both Oracle Configurator and Oracle Advanced Pricing.) For more information, see:

- Chapter 10, "Set Up Configurator and Customize the Solution"
- Chapter 11, "Set Up Configurator Functional Companions"
- Chapter 12, "Set Up Quoting"

9.1 Set Up One-Time Charges

To setup a One-Time Charge you must:

- Create One-Time Charge Names
- Define Modifier for an Item or Service
- Define Qualifiers on the Modifier

9.1.1 Create One-Time Charge Names

You must create names for one-time charges--such as Change Fee, Install Fee and Disconnect Fee--as a Lookup of type FREIGHT_CHARGE_TYPE.

Module

Oracle Applications (Forms)

Responsibility

Oracle Pricing Manager

Navigation

Setups > Lookups > Oracle Pricing Lookups page

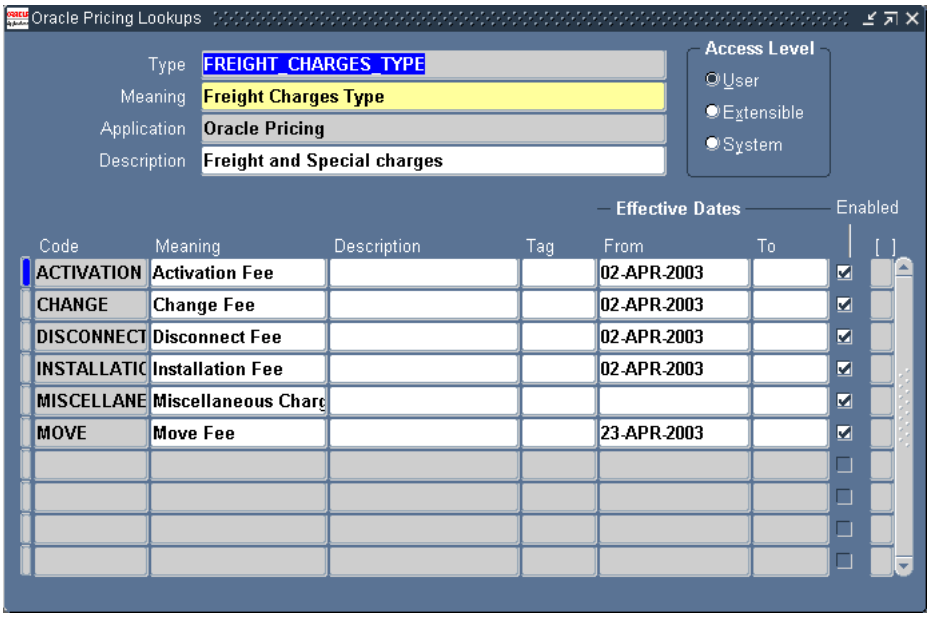
Steps

1. On the Oracle Pricing Lookups page, search for the Lookup Type **FREIGHT_CHARGES_TYPE**.
2. Enter a Code, Meaning and Description for the one time charge.

   Note that what you enter in the MEANING field is what appears in the One-Time Charges column in the Oracle Quoting Lines and Overview pages.
3. Select the Enable option.
4. Save your work.

   Figure 9–1 shows a sample setup of the Lookup Type FREIGHT_CHARGES_TYPE.

Figure 9–1 Lookup Type FREIGHT_CHARGES_TYPE



9.1.2 Define Modifier for an Item or Service

Create a modifier to specify the price of the One-Time Charge for a selected item.

Module

Oracle Applications (Forms)

Responsibility

Oracle Pricing Manager

Navigation

Modifiers > Modifier Setup > Advanced Pricing - Define Modifier page

Steps

1. On the Advanced Pricing - Define Modifier page, click the Main tab.
2. On the Main tab, select a modifier **Type = Freight and Special charge list**.
3. Enter a Number for the modifier.

This number can be any text or numeric value, such as, **Change Service**.

4. Enter a Name for the modifier.

   This name can be any text value, such as, **Change Service**.

5. Select the Currency for the modifier and specify a Description.

6. Select the Active option.

7. On the Modifiers Summary tab:

   – If you do not enter a **Modifier No** (this can be a text value), a number automatically generates for you.

   – Select the **Level = Line**, since the charge applies at the Line Level.

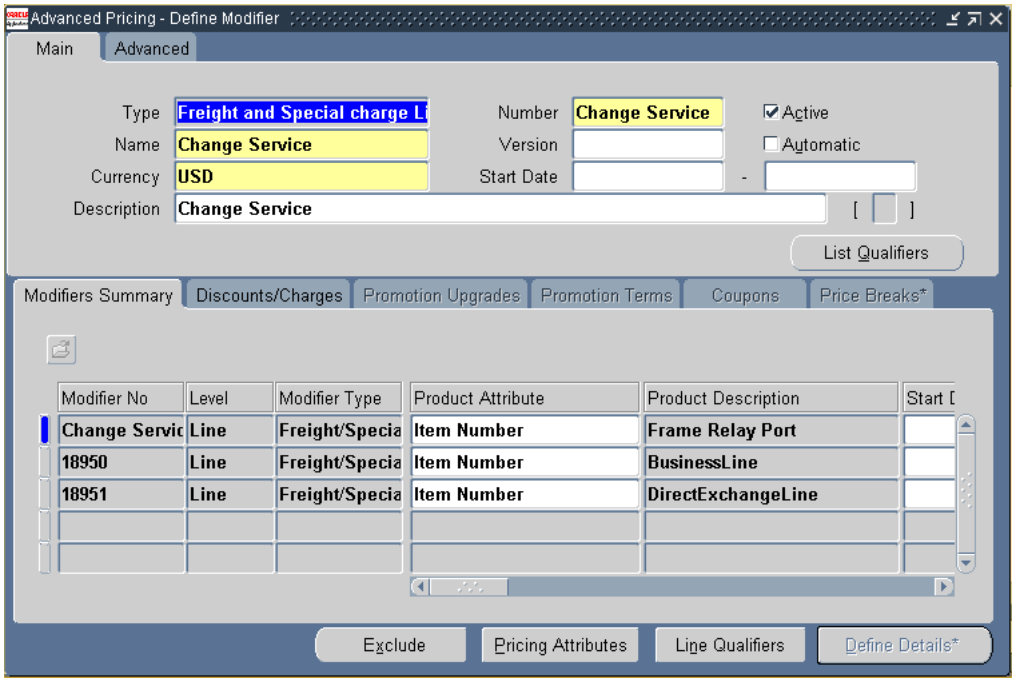
     For example, a Change Fee applies at the Line level.

   – Select **Modifier Type = Freight/Special Charge**.

   – Set **Product Attribute = Item Number**, **Product Attribute Value = <Item Number>**, and select a precedence.

   Figure 9–2 shows a sample window of Modifiers Summary tab of the Advance Pricing - Define Modifier page.

Figure 9–2 Modifiers Summary Tab



8. On the Discounts/Charges tab:
   - Select the Charge Name.

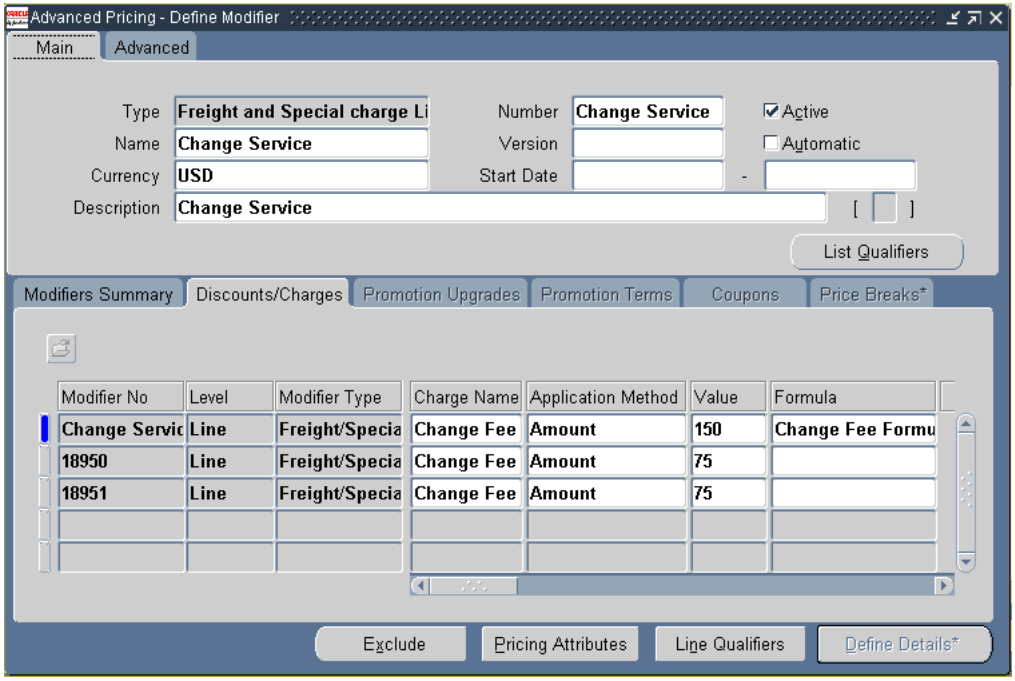
     This is the Lookup of Type **FREIGHT_CHARGES_TYPE** that you created previously.
   - Select and Application Method.

     For more information, refer to *Oracle Advanced Pricing Implementation Guide*. This example uses the Application Method of **Amount**.
   - Enter a Value.

     This is the value of the modifier, such as the price of the Change Fee for the Frame Relay Port item.
9. Save your work.

   Figure 9–3 shows a sample of Discounts/Charges tab of the Advanced Pricing - Define Modifier page.

Figure 9–3 Discounts/Charges Tab



9.1.3 Define Qualifiers on the Modifier

You use qualifiers to specify when to apply a one-time charge to an item.

The setup example shows the creation of a qualifier so that the one-time Change Fee applies to the item only when the provisioning action (which is identified by the Line Type of the item) on the item is CHANGE.

Module

Oracle Applications (Forms)

Responsibility

Oracle Pricing Manager

Navigation

Modifiers > Modifier Setup > Advanced Pricing - Define Modifier page

Steps

1. On the Advanced Pricing - Define Modifier page, select the Charge No to which the Qualifier is to be defined.

   The setup example shows the selected **Change Service** modifier.

2. Click the Line Qualifiers button.

   This causes the Line Level Qualifiers page to appear.

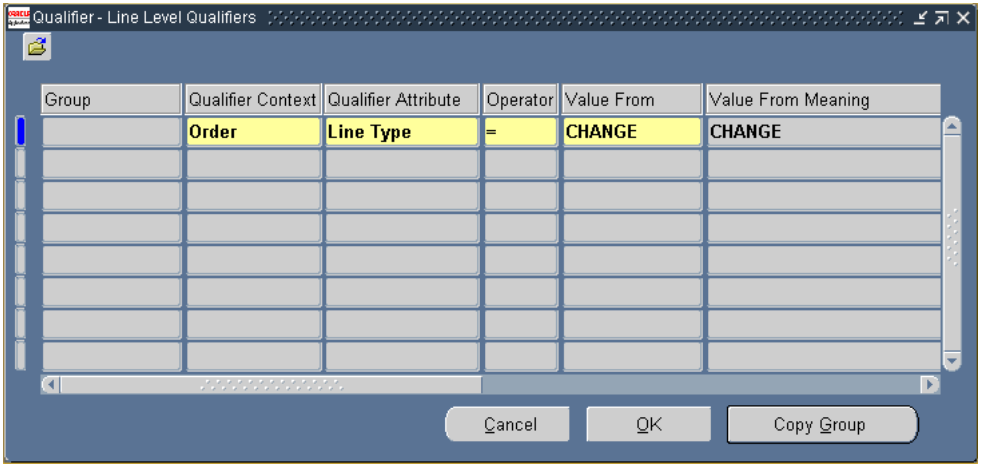
3. Choose or specify:

   - Qualifier Context
   - Qualifier Attribute equal to **Line Type**
   - Operator equal **=**
   - Value From equal to **CHANGED**

     This is the line type defined in Order Management.

   Figure 9–4 shows sample data in the Qualifier - Line Level Qualifiers window.

Figure 9–4 Qualifier - Line Level Qualifiers Window



9.2 Discount to One-Time Charges

Setup the Order Capture descriptive flexfield, **LINE: Additional Information**, to capture the line charge discount percent.

Define a formula with the **Get_Custom Price** function to query the descriptive flexfield attribute and apply it to the amount specified for the One Time Charge in the modifier form.

9.3 Pricing Attributes and Sourcing Rules

Pricing objects in Oracle Advanced Pricing enable users to define pricing actions and pricing rules for a given business process. Through price lists, modifiers, and formulas, these pricing actions provide the ability to define prices, price adjustments, and other benefits. By defining pricing rules such as qualifiers and pricing attributes, you use these pricing rules to drive your pricing actions. You can use the attributes as elements to express your product or customer hierarchy.

A product hierarchy element is a level at which you can define pricing characteristics. Examples of product hierarchy include item number, category, or brand. Customer hierarchy is an example of a qualifier hierarchy, such as customer group.

Through Attribute Management, the Pricing Engine receives all the values of the attributes defined in the qualifier and pricing attributes in order to determine which price lists and modifiers the transaction is qualified for. The data sources for the qualifiers and pricing attributes can be within Oracle Applications or from outside Oracle Applications. Users can use the feature of Attribute Mapping to extend Pricing to tap into data from a wide variety of non-standard sources to drive their pricing actions. In the attribute management windows, users can create new attributes, update existing attribute properties, or disable attributes.

The calling application first uses the attribute mapping API to get all qualifier and pricing attributes associated with the transaction. Then the calling application makes a call to the pricing engine. The pricing engine evaluates these values to determine which price lists and modifiers are eligible for the transaction. The data sources for the qualifiers and pricing attributes can be within Oracle Applications or from outside Oracle Applications.

The Attribute Management feature enables you to create new contexts and attributes, update existing context or attribute properties, or disable existing contexts. Creating new context or attributes extends Oracle Advanced Pricing's ability to provide user defined data sources to drive pricing actions. The methods to source data for an attribute are:

- User Entered (attribute value is entered by user)
- Custom Sourced (custom code is used to derive an attribute)

- Attribute Mapping (Pricing Engine derives information from other Oracle Applications and non-Oracle data sources.

In the context of TSO, attribute management can be used to derive the price of a Frame Relay PVC depending upon the PVC Speed which is both an Install Base attribute and a Configurator Attribute. You can custom-source the value of this attribute and provide it to the pricing engine to derive the price based on this attribute value.

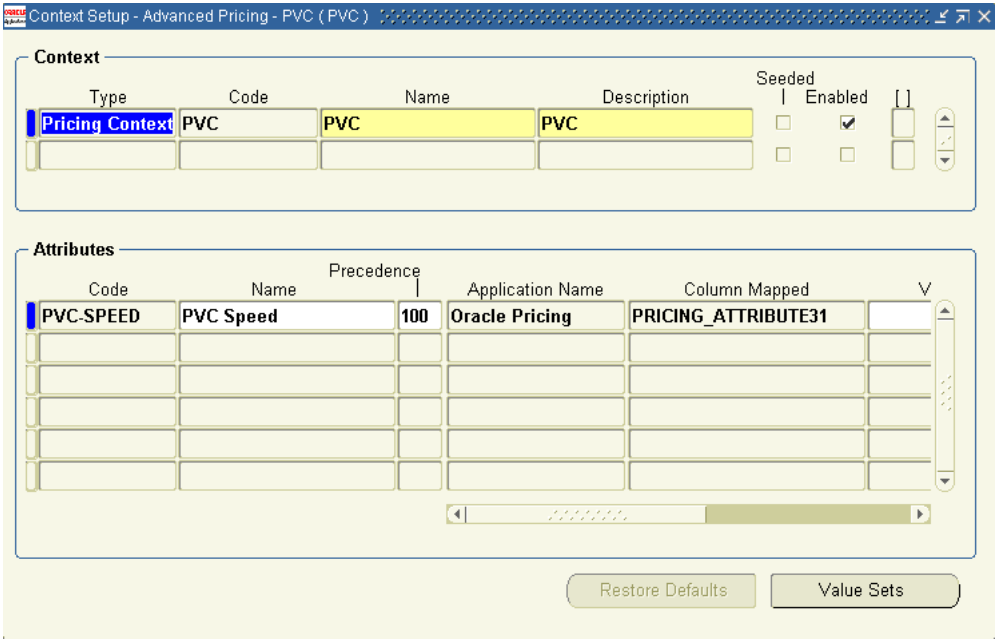
Attribute Based Pricing Setup

The navigation to Attribute Based Pricing setup is:

Pricing > Setup > Attribute Management > Context and Attributes

Figure 9–5 shows an example of setting up pricing attributes for PVC Speed.

Figure 9–5 Context and Attributes



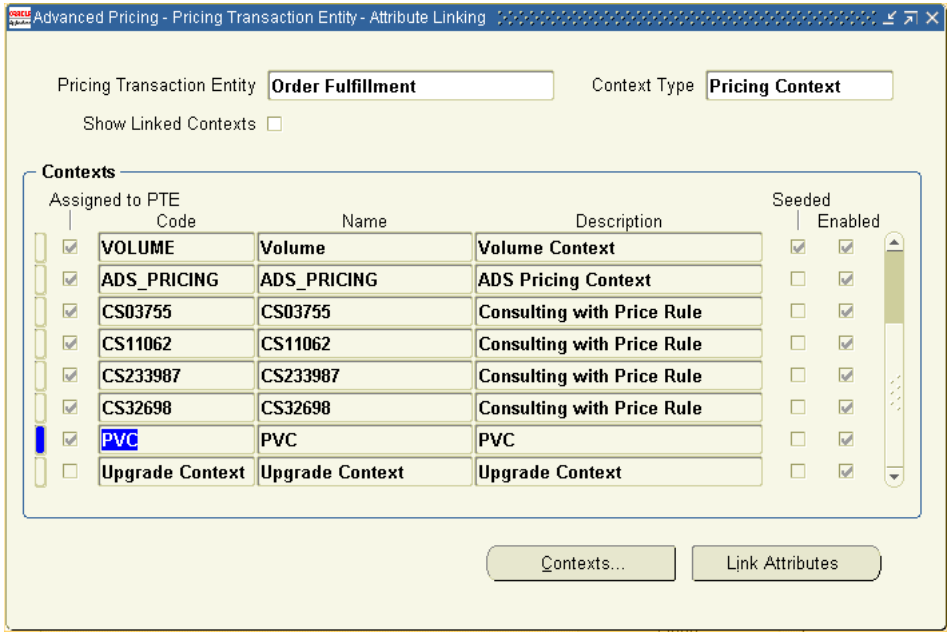
Attribute Linking and Mapping

The navigation to Attribute Linking and Mapping setup is:

Pricing > Setup > Attribute Management > Attribute Linking and Mapping

Figure 9–6 shows an example of pricing attribute linking and mapping for PVC.

Figure 9–6 Attribute Linking and Mapping



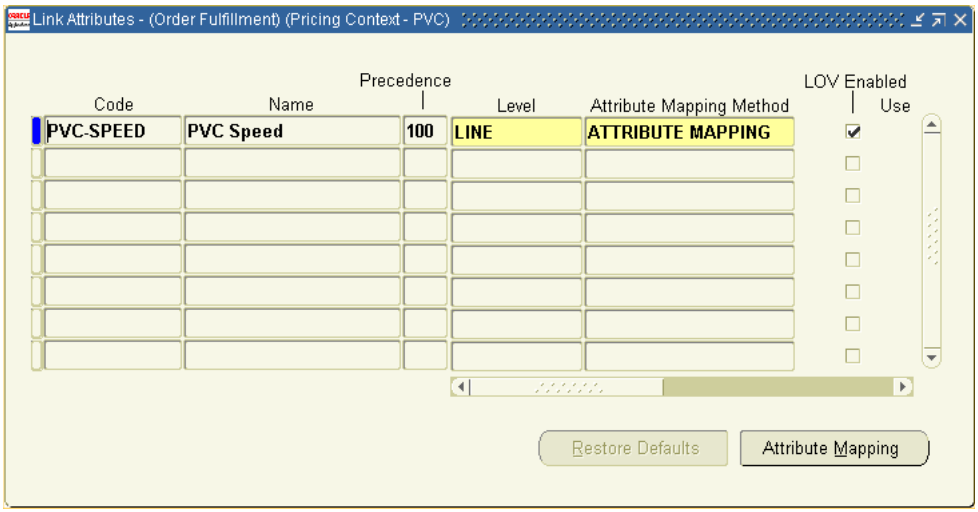
Attribute Linking

The navigation to Link Attributes setup is:

Pricing > Setup > Attribute Management > Attribute Linking and Mapping > Link Attributes

Figure 9–7 shows an example of the attribute linking.

Figure 9–7 Attribute Linking



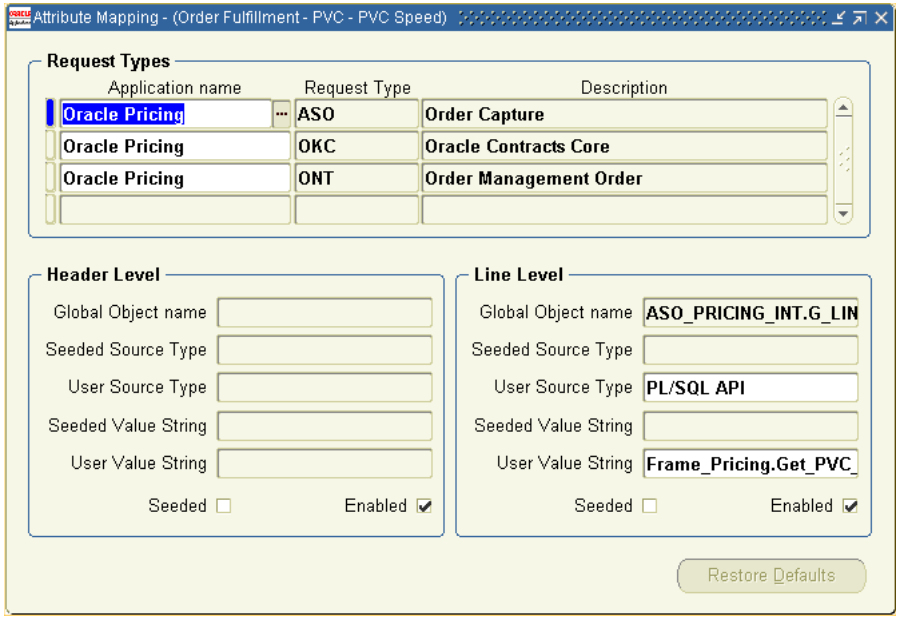
Attributes Mapping

The navigation to Attributes Mapping is:

Pricing > Setup > Attribute Management > Attribute Linking and Mapping > Link Attributes > Attributes Mapping

Figure 9–8 shows an example of attribute mapping.

Figure 9–8 Attribute Mapping



The PLSQL Package can use the elements of the global object ASO_PRICING_INT.G_LINE_REC as input paramters.

For example, the global paramter quote_line_id can be used to link via the configurator keys to the CZ_Config_Ext_Attributes table to select the attribute value for by the attribute name.

The value returned is sourced to the Pricing Attribute PVC_Speed. This pricing attribute can either be used in a Formula and associated to a price list or can be just used to define various prices for an item.

9.4 Get Custom Price

This is a public, supported API that lets you extend Oracle Advanced Pricing to obtain a price lets you extend the functionality of the product while avoiding the downsides of customization. The data structure underlying Advanced Pricing may change over time, but the mechanics of **Get_Custom_Price** will not (or will be migrated if they do). In other words, if you keep all your custom code in this function, you are playing by the rules.

Get_Custom_Price works in conjunction with Formulas. To use Get_Custom_Price, you first define a formula in Oracle Advanced Pricing and source a formula line as a Function. You use the **formula_id** attribute value in **Get_Custom_Price** to determine which block of code to execute for which formula.

It is important to note that during installation of Oracle Advanced Pricing, only the package header is created. It is your responsibility to create the package body function called **Get_Custom_Price**. This API has the following set of parameters:

```
P_price_formula_id: IN NUMBER
P_list_price: IN NUMBER
P_price_effective_date: IN DATE
P_req_line_attrs_tbl: IN QP_FORMULA_PRICE_CALC_PVT.REQ_LINE_
ATTRS_TBL)

P_price_formula_id identifies the formula from which the API is called.
P_req_line_attrs_tbl provides information such as product, pricing attributes
and qualifiers.
P_req_line_attrs_tbl contains the following fields:
   Line_index: Line index of the price request line
    Attribute_type: Qualifier, product, pricing
    Context: Context name (for example, Item)
    Attribute: Attribute name (for example, Pricing_attribute1)
    Value_from: Attribute value
```

For more information refer *Oracle Advanced Pricing Implementation Guide*.

10

Set Up Configurator and Customize the Solution

The main topics in this chapter are:

- Section 10.1, "About Configurator Implementation in the Solution" on page 10-3
- Section 10.2, "Create the Configuration Model" on page 10-16
- Section 10.3, "Publish a Container Model" on page 10-18
- Section 10.4, "Define Profile Options" on page 10-18
- Section 10.5, "Disable Pricing" on page 10-19
- Section 10.6, "CZ Schema Customizations" on page 10-19
- Section 10.7, "Initialization Parameters" on page 10-27
- Section 10.8, "Batch Validation Parameters" on page 10-29
- Section 10.9, "Programmatic Tools for TSO Development" on page 10-30

Reference Material

This chapter describes setup tasks that you perform to Oracle Configurator that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on functionality, setting up, or using Oracle Configurator, refer to:

- Section 3.4, "Configurator" on page 3-7
- *Oracle Configurator Developer User's Guide*
- *Oracle Configurator Implementation Guide*
- *Oracle Configurator Installation Guide*
- *Oracle Configuration Interface Object (CIO) Developer's Guide.*

- *Oracle Configurator Methodologies.*
- *Oracle Configurator Release Notes*

Before You Begin

Before you can set up Oracle Configurator for use with the Oracle Telecommunications Service Ordering solution, you must set up Oracle Bill of Materials, Oracle Install Base, and Oracle Order Management. For more information, see:

- Chapter 7, "Set Up Bill of Materials"
- Chapter 8, "Set Up Install Base"
- Chapter 5, "Set Up Order Management"

In addition to this chapter, there is another chapter on setting up Configurator for use with the Telecommunications Services Ordering solution, Chapter 11, "Set Up Configurator Functional Companions".

After You Set Up Oracle Configurator

After you set up Oracle Configurator, you can set up Oracle Quoting. Note, however, that Quoting is also dependent on the setup of Oracle Advanced Pricing. For more information, see Chapter 12, "Set Up Quoting".

10.1 About Configurator Implementation in the Solution

Topics in this section include:

- Section 10.1.1, "Conventions for Connector and Components of a Model" on page 10-3
- Section 10.1.2, "Using Container Models" on page 10-4
- Section 10.1.3, "Container Model Settings and Structure" on page 10-4
- Section 10.1.4, "Importing Container Models into Oracle Configurator" on page 10-5
- Section 10.1.5, "Structuring Container Models" on page 10-7
- Section 10.1.6, "Connecting Components" on page 10-11
- Section 10.1.7, "Using Configuration Rules" on page 10-13

10.1.1 Conventions for Connector and Components of a Model

Some of the diagrams that appear in this chapter show the structural relationship between connectors and components of a Model. The following sample diagram and Table 10–1 show and describe the conventions for the Model diagrams.

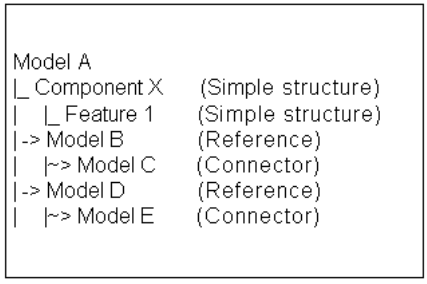


Table 10–1 refers to the preceding diagram.

Table 10–1 Model Diagram Conventions

Diagram Part	Description
Model A |_ Component X	This part represents simple Model structure. In this case, Component X is a child ofModel A.
| |_Feature 1	Feature 1 is a child of Component X.

Table 10–1 Model Diagram Conventions

Diagram Part	Description
|->Model B	`This represents a Reference node. Model A references Model B.`
|~> Model C	This represents a Connector node. Model B has a Connector to Model C.

10.1.2 Using Container Models

You can mark some BOM items, such as a BOM Model, Option Class, or standard item, within a Container Model as trackable. Tracking enables Oracle Install Base to record information about the item, including its location, status, current configuration, change history, and so on. You must define the Container Model itself as untrackable. Oracle Install Base does not record information about untrackable items. For more information about making an item Install Base trackable, see Section 6.2, "Specify Install Base Tracking for Each Component Item" on page 6-3.

Before an end user can update any of a Container Model's installed configurable instances, the Container Model must meet certain requirements, such as a valid Container Model structure, Connectors, and rules, to ensure that the configured items are trackable in Oracle Install Base. For more information about the necessary requirements, see:

- "Container Model Settings and Structure" on page 10-4
- Section 6.5, "Create a Container Model" on page 6-6

10.1.3 Container Model Settings and Structure

The following list identifies the requirements for a Container Model in Oracle Inventory and Oracle Bills of Material.

A Container Model **must** be:

- A Pick-to-Order (PTO) BOM Model

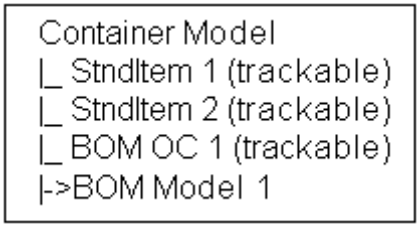
A Container Mode **cannot**:

- Be trackable.
- Reference another Container Model. In other words, a Container Model cannot be a component within another Container Model.

- Contain more than one effective reference to the same trackable BOM Model For more information about references and effectivities, see the *Oracle Configurator Developer User's Guide*.
- Contain a non-trackable BOM Model with trackable children--one or more BOM Option Classes or BOM Standard Items.
- Contain any BOM Models or BOM Option Classes that have a default quantity greater than 1.
- Have any trackable BOM Standard Items or trackable BOM Option Classes as direct children.

  For example, Figure 10–1 on page 10-5, shows BOM trackable items StndItem1, StndItem2, and BOM OC 1. This structure is invalid because these BOM items are all direct children of the Container Model.

Figure 10–1 Invalid Container Model Structure



For more information on creating or specifying Container Models, see Section 6.5, "Create a Container Model" on page 6-6.

10.1.4 Importing Container Models into Oracle Configurator

To develop a Container Model in Oracle Configurator Developer, you must import the Oracle Inventory and BOM data into the Oracle Configurator (CZ) schema of the Oracle Applications database. For information about populating the CZ schema, see the *Oracle Configurator Implementation Guide*.

The concurrent programs under Populate and Refresh Configuration Models in Oracle Applications import the completed bills of material into the Oracle Configurator schema. Run the concurrent programs from the Oracle Configurator Administrator menu.

These concurrent programs import BOM structure--assemble-to-order, pick-to-order or Container Models, structure, and rules--and require complete BOM that are identified at the desired root.

The concurrent programs under Populate and Refresh Configuration Models in Oracle Applications include:

- **Populate Configuration Models**: Populates the CZ schema online tables with data that is appropriate for creating configuration models that are based on existing BOM or legacy data.
- **Refresh a Configuration Model**: Updates a specific BOM Model with any changes that may have been made in Oracle Applications since the time the BOM was first imported into the CZ schema.
- **Refresh all Imported Configuration Models**: Updates all of the BOM Models that were imported into the specific database instance that the user is logged into.
- **Disable/Enable Refresh of a Configuration Model**: Enables the user to prevent specific configuration models from being refreshed if the Refresh all Imported Configuration Models concurrent program is run.

When importing a container BOM Model, the system creates a Model node in Oracle Configurator Developer. If the imported BOM Model has components that are assemble-to-order or pick-to-order Models, then the import process creates:

- A model structure for each child Model.
- A corresponding reference node in the parent model.

Important! The concurrent programs under Populate and Refresh Configuration Models do *not* verify that all of the settings you selected when defining a Container Model in Oracle Applications are correct. If a Container Model violates any of the requirements listed in this section, an error occurs only when you generate the Active Model in Configurator Developer, not when you import or refresh the model.

The Refresh a Configuration Model and the Refresh all Imported Configuration Models concurrent programs check only two settings that you specify in Oracle Applications. When you refresh a Container Model, the concurrent program displays:

- A warning, if the **Configuration Model Type** changed from **Container** to **Standard**. The **Refresh** concurrent programs do not display a warning if you changed the **Configuration Model Type** from **Standard** to **Container**.
- An error, if any of the Container Model's child BOM Models capable of multiple instances changed from a pick-to-order to an assemble-to-order BOM Model. For example, the **Instances Minimum** and **Maximum** values for Model X (a pick-to-order BOM Model) are not equal in Configurator Developer. In Oracle Inventory, you change Model X from a pick-to-order model to an assemble-to-order model. When you refresh the Container Model, the concurrent program displays an error.

10.1.5 Structuring Container Models

In a Container Model, if one of the trackable child Models does not have a trackable parent or ancestor, then there cannot be a limit on how many instances of that trackable child Model that can be created at runtime. In other words, the trackable BOM Models Instances Minimum and Maximum values must be 0/null in Configurator Developer.

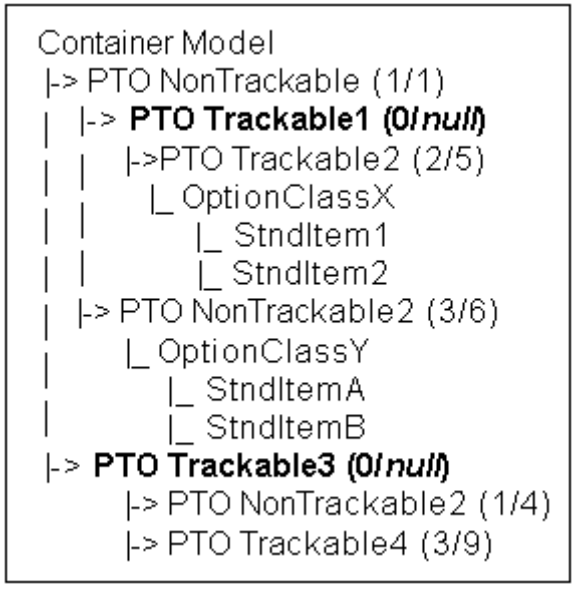
If the Instances Minimum and Maximum values are not **0/*null***, respectively, for a **trackable instance**, Configurator Developer displays an error when you generate the **Active Model**. A trackable instance can contain both trackable and non-trackable items, such as BOM Models and Option Classes, as well as components created in Configurator Developer.

Note: The first time you import a Container Model, the Populate Configuration Models concurrent program sets the Instances Minimum and Maximum values to **0/*null***, respectively, for all trackable child BOM Models. However, refreshing a Container Model does not reset these values if you change them in Oracle Configurator Developer. If the model's structure violates any of the requirements listed in this section, Oracle Configurator Developer displays an error when you generate the Active Model.

Because you can create multiple instances of such a BOM Model during configuration and Oracle Install Base tracks these instances when they are installed, a BOM Model matching this criteria is a trackable instance. An example of a trackable instance appears in Figure 10–2.

Note: Remember that you can only track BOM items and that Oracle Install Base only records information about trackable BOM items.

Figure 10–2 Example of Trackable Instances



In Figure 10–2, PTO Trackable1 and PTO Trackable3 (in bold) are trackable instances because they are trackable and do not have a trackable parent or ancestor. Therefore, you must set the Instances Minimum and Instances Maximum values to **0** and ***null***, respectively. The table shows these settings as **0/*null***.

Note that the Instances Minimum and Instances Maximum values for PTO Trackable 2 and PTO Trackable 4 are not set to **0/*null***. This is because a trackable model that is a child or a descendant of a trackable instance does not have the same configuration requirements as a trackable instance. That is, the trackable child instance does not have to obey the rule when no instances exist when both:

- The configuration session begins.
- There is no limit to how many instances the end user can create.

Refer to the *Oracle Configurator Developer User's Guide* for more information about creating multiple instances of components during configuration.

Examples of Valid and Invalid Container Model Structure

This section provides examples of how trackable instances can impact the validity of a Container Model's structure in Oracle Configurator Developer.

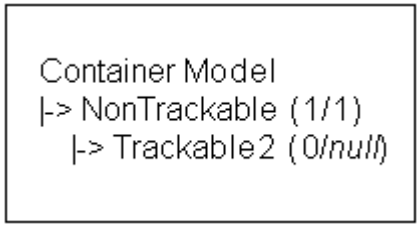
In a Container Model, a trackable descendant model can be a child of a non-trackable model only if the non-trackable model's Instances Minimum and Instances Maximum values are both set to 1, and:

- The trackable model's Instances Minimum and Instances Maximum values are **0/*null***. See Figure 10–3.

  or

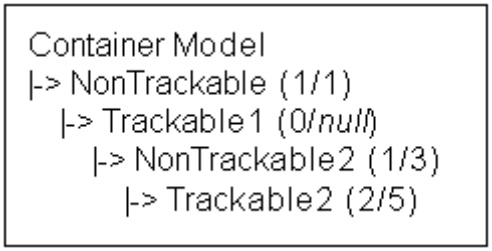
- Another trackable model whose Instances values are **0/*null*** is the parent of the non-trackable model. See Figure 10–4.

Figure 10–3 Valid Model Structure: Non-Trackable Parent and Trackable Child



In Figure 10–3, the model structure is valid because the Instances Minimum and Maximum for Trackable2 equals **0/*null***.

Figure 10–4 Valid Model Structure: Trackable Child and Trackable Ancestor



Although Trackable2 is a child of NonTrackable2, the structure shown in Figure 10–4 is valid because the Instances Minimum and Instances Maximum for Trackable1 equal **0/*null*** and Trackable1 is an ancestor of Trackable2. Because Trackable1 is a trackable instance, an Oracle Install Base or Oracle Quoting user can select an installed instance of Trackable1 and launch Oracle Configurator to update it.

Figure 10–5 Invalid Model Structure: Non-Trackable Parent and Trackable Child

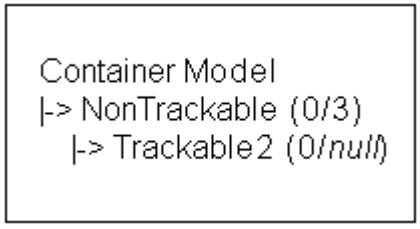
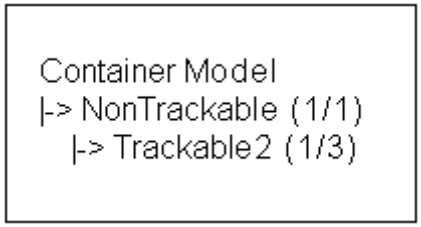


Figure 10–5 shows an invalid structure because NonTrackable's Instances Minimum equals 0. If an Oracle Install Base or Oracle Quoting user selects an instance of Trackable2 for update, its parent (NonTrackable) is not part of the configuration when the Oracle Configurator session begins. Therefore, Oracle Configurator cannot add an instance of Trackable2 to the configuration either.

Figure 10–6 Invalid Model Structure: Non-Trackable Parent and Trackable Child



In Figure 10–6, Trackable2 is the first trackable BOM Model in the Container Model's structure. However, its Instances values are not **0/*null***. This structure is invalid because although you can configure and track Trackable2 in Oracle Install Base, there is a restriction on how many instances of Trackable2 are allowed at runtime.

10.1.6 Connecting Components

Oracle Install Base tracks the relationships that the connections formed that you create between components. Oracle Install Base keeps a record only of *trackable* BOM items, so any connections that you make at runtime must be from an instance of a trackable BOM Model to instances of other trackable BOM Models.

Because Oracle Install Base keeps a record of connected instances only if they are trackable, any Connectors that you create within a Container Model in Oracle Configurator Developer must be valid. If a Container Model contains Connectors that violate any of the requirements listed in this section, Oracle Configurator Developer displays an error when you generate the Active Model.

Connectors enable you to define rules that include nodes from trackable instances as participants. See Section 10.1.7, "Using Configuration Rules" on page 10-13.

When working in a Container Model in Oracle Configurator Developer, you must create Connectors only from:

- A trackable BOM Model to another trackable BOM Model
- A non-trackable BOM Model to another non-trackable BOM Model

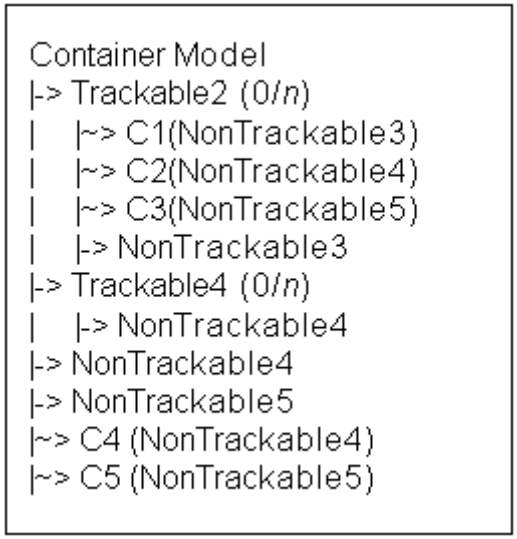
You must *not* create Connectors:

- From the Container Model's root node to a trackable model.
- From the Container Model's root node to any model that is a child of a trackable model.

For example, a Container Model references a trackable model named PTO1. PTO1 references a non-trackable model named PTO2. You create a Connector from the Container Model to PTO2. Because you created a Connector from the Container Model to a model that is a child of a trackable model, Oracle Configurator Developer displays an error when you generate the Active Model.

- Within a trackable model to a non-trackable model that has the trackable model as an ancestor (see Figure 10–7 on page 10-12).

Figure 10–7 Invalid Connectors



In Figure 10–7:

- Connector C1, C2, and C3 are invalid because you cannot connect a trackable model and a non-trackable model.
- Connectors C4 and C5 are allowed because the Connectors' parent and target are both non-trackable models (in other words, they do not cause an error when you generate the Active Model). However, you cannot connect NonTrackable5 to either of the specified targets because the targets' parent (Trackable2) is trackable.

Note: In Oracle Configurator Developer, when you select a BOM Model node, the **BOM Item Type** field indicates whether it is an assemble-to-order model, a pick-to-order model, or a Container Model. Additionally, when you select any BOM node, the **Trackable** check box indicates whether the item is trackable. You can expand the **Definition** attribute to view these settings.

10.1.7 Using Configuration Rules

Oracle Quoting and Oracle Install Base end users typically need to update only a few components within a network of connected components. To facilitate this process and improve runtime performance, you can view only the following components when an Oracle Configurator session begins:

- The component(s) selected for update.
- Any components required to enforce constraints defined in the model.

When an end user updates an installed configuration, Oracle Configurator may need to add components to the configuration session to ensure the validity of all connections, and the entire configuration. At runtime, only rules that you define using Connectors can propagate to component instances that may not be visible in the configuration session.

For this reason, it is important to define rules using Connectors when any rule participants are not part of the same trackable instance. In other words, no parent-and-child or ancestor-and-descendant relationship exists between the trackable items. For more information on trackable instances, see Section 10.1.5, "Structuring Container Models" on page 10-7.

Note: If two trackable items are part of the *same* trackable instance, it is not necessary to create a Connector between them in order to define a rule in which they are both participants. For example, Model A is a trackable BOM Model that has two children, Model B and Model C. Since Model A references these models (thus they are Model A's children), you can use nodes from both Model B and Model C when defining rules.

When a rule defined using Connectors propagates to instances that are not visible in the configuration session, Oracle Configurator prompts you to add instances to the

configuration session so other required changes are applied and the configuration remains valid.

For more information on using Connectors to define rules, see the *Oracle Configurator Developer User's Guide*.

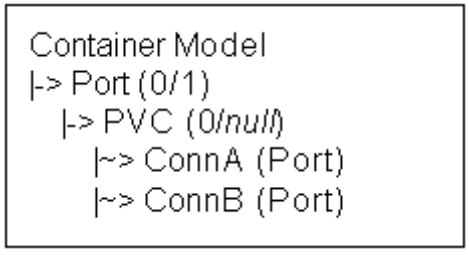
Valid Configuration Rule Combinations

The following are examples of valid configuration rules for trackable models within a Container Model.

- A rule can contain participants from a trackable model and one of its trackable child Models.
- You can create a rule between Connectors that have the same trackable model as their target.

  Using the structure in Figure 10–8 on page 10-14 as an example, you can create rules using nodes from Connector A and Connector B, even though their targets are the same. (You can also create rules using nodes from Connector A and Connector B if their targets are different.)

Figure 10–8 Connectors to the Same Trackable Model



- A rule that includes nodes from two non-trackable models is valid as long as neither model is a child of a trackable model, or they are both children of the same trackable model.

Invalid Configuration Rule Combinations

If a configuration rule violates any of the requirements in this section, Oracle Configurator Developer displays an error when you generate the Active Model.

The following are examples of **invalid** configuration rules for trackable models within a Container Model.

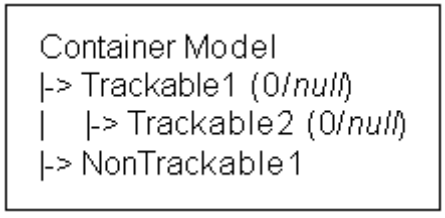
- A numeric rule that dynamically changes how many allowable instances of a **trackable** child BOM Model can exist at runtime. This type of rule changes how many instances of a model are allowed at runtime.
- A numeric rule in which a Container Model node contributes to or consumes from the minimum or maximum number of allowed instances at runtime for another model (regardless of the model's type).

  For more information about this type of numeric rule, see the *Oracle Configurator Developer User's Guide*.
- A rule that uses nodes from sibling trackable models (that is, the models exist at the same level in the Container Model's structure).
- A rule between a trackable model and a non-trackable model that is not one of the trackable model's children.

  For example, using the structure shown in Figure 10–9 on page 10-15, create a rule containing participants from Trackable1 and NonTrackable1. This rule is invalid and causes an error when you generate the Active Model.

  Note: This restriction does not apply to Functional Companions.

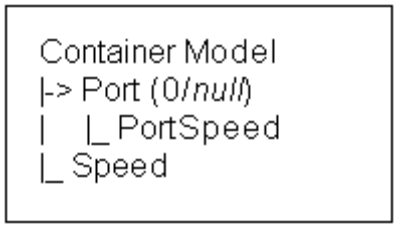
Figure 10–9 Container Model Structure0



- A rule with participants from one trackable model and any other node from a non-trackable child of the Container Model.

  For example, in the structure appearing in Figure 10–10 on page 10-16, both PortSpeed and Speed are Features. A rule using PortSpeed and Speed is invalid because Port is trackable, PortSpeed is a Feature in Port, and Speed is a direct child of the Container Model.

Figure 10–10 Container Model Structure



10.2 Create the Configuration Model

This section explains how to define a configuration model specifically for an Oracle Telecommunications Service Ordering solution.

For information about defining configuration models, see the *Oracle Configurator Developer User's Guide*.

Module

Forms (Oracle Applications)

Responsibility

Configurator Administrator

Navigation

Populate/Refresh Configuration Models > Populate Configuration Models

Steps

1. Import the Container Model by running the Import Configuration Models concurrent program.

   For more information on:

   – Setting up Container Models, see Section 6.5, "Create a Container Model" on page 6-6.

   – Concurrent programs, see the *Oracle Configurator Implementation Guide*.

2. After importing the Container Model, verify that its structure is valid by opening the Model for editing in Configurator Developer, and choosing **Tools > Generate Active Model**.

3. Add the Model structure and define configuration rules.

For more information, see the *Oracle Configurator Developer User's Guide.*

4. Generate a User Interface and optionally create Activate Instance UI controls on specific UI windows.

   An Activate Instance UI control is a button or picture to which you assign the Activate Instance action.

   To generate a new User Interface, see the *Oracle Configurator Developer User's Guide*.

   Customizing the User Interface consists of adding Activate buttons, creating Connection List buttons, hiding Functional Companions, and altering the format in the UI Editor.

   To add graphics and buttons to specific UI windows, see the *Oracle Configurator Developer User's Guide*.

   Note: To help end users to navigate when updating an installed configuration, you may also want to display the Navigation Tree in a Container Model's User Interface. For more information, see the *Oracle Configurator Developer User's Guide*.

   The Activate Instance UI control appears in a UI window only if you create the control on the selected component's UI window. When you make the Oracle Configurator UI window editable, the Activate Instance control does not appear.

5. Define Line Types in Oracle Order Management (OM).

   In Order Management, Transaction Types are the source of various attributes on a sales order's header and lines.

   When you update an installed configuration, Oracle Configurator uses **Line** transaction types to describe the type of change made to each item. For more information, see:

   – *Oracle Order Management Suite Implementation Manual*

   – *Oracle Order Management User's Guide*

6. Adapt the following required Functional Companions:

   – **Line Type**: Populates the Line Type column in the Summary window. This column lists the type of change made to each item during the update

configuration session. For more information, see Section 11.1.3, "Specify the Line Type" on page 11-9.)

- **Location**: Sets the Location ID and Location Type Code for BOM items and passes this information to downstream applications (that is, Order Management, Service Fulfillment Manager and Install Base). For more information, see Section 11.1.2, "Specify the Default Location" on page 11-5.

7. Optionally, adapt the following Functional Companions:
   - **Instance Name**: Enables an end user to modify the names of component instances at runtime. For more information, see Section 11.1.4, "Change the Instance Name" on page 11-18.
   - **IBAttribute**: Collects configuration attributes of configured components. For more information, see Section 11.1.5, "Using Configuration Attributes with Install Base (IB)" on page 11-21.
8. Publish the Container Model and specify Oracle Install Base, Oracle Quoting, and Oracle Order Management in the list of hosting applications.

   For a list of Oracle Applications short names, see the *Oracle Configurator Implementation Guide*. For information about publishing, see the *Oracle Configurator Developer User's Guide*.

10.3 Publish a Container Model

To allow Oracle Install Base or Oracle Quoting users to update installed configurations, you must include Install Base and Quoting when publishing the Model from Configurator Developer.

Additionally, you must make the publication available to Oracle Order Management. For more information about publishing, see the *Oracle Configurator Developer User's Guide*.

10.4 Define Profile Options

You must define some profile options before Install Base and Quoting users can update installed configurations using Oracle Configurator.

The following Oracle Configurator profile options affect how Oracle Configurator integrates with Install Base:

- CZ: Configurator Install Base
- CZ: Suppress Baseline Errors

- CZ: Report All Baseline Conflicts

These profile options have default values, but you may want to modify them for your installation. For more information, see the *Oracle Configurator Installation Guide*.

You must also define some additional Oracle Install Base and Oracle Quoting profile options to allow these applications to integrate with Oracle Configurator. For more information, refer to:

- *Oracle Quoting Implementation Guide*
- *Oracle Install Base Implementation Guide*

10.5 Disable Pricing

Pricing attributes sourced from Configurator output attributes are not available to the pricing engine until you have exited the Configurator by clicking Done. Therefore, if you are sourcing pricing attributes from Configurator output attributes, you should disable pricing display within the Configurator.

If your application that supports updates to installed configurations includes pricing that depends on the Line Type (action) or attributes of the line item, then you should disable pricing. Since such a dependency is likely to be common, it is recommended that you disable pricing in Oracle Configurator if your application supports updates to installed configurations.

To disable pricing in Oracle Configurator, set the `cz.activemodel` property of the OC Servlet as follows:

```
cz.activemodel=/nolp|/nodp|
```

For more information on setting this property, see the *Oracle Configurator Installation Guide*.

10.6 CZ Schema Customizations

This section describes some required customizations to the CZ schema that you must make for the TSO solution:

- Section 10.6.1, "TSO Solution CZ_DB_SETTINGS Parameters" on page 10-20, for information on a setting that controls the display of the Line Type column in the Summary window.
- Section 10.6.2, "CZ_CONFIG_ATTRIBUTES Table" on page 10-21, for a description of the table used to hold configuration attribute data.

- Section 10.6.3, "CZ_CONFIG_EXT_ATTRIBUTES Table" on page 10-21, for a description of the table used to hold configuration attribute data while it is being compared with the data in Oracle Install Base
- Section 10.6.5, "MACD_FULFILLMENT_STRUCTURES Table" on page 10-23, for instructions on how to create the table needed by the Line Type Functional Companion.
- Section 10.6.6, "Mapping of CZ tables to Install Base Schema (CSI)" on page 10-25, for background information that helps you understand how data in the CZ schema is related to data in Install Base.

10.6.1 TSO Solution CZ_DB_SETTINGS Parameters

See the *Oracle Configurator Implementation Guide* for background on the CZ_DB_SETTINGS table.

Table 10–2 Settings in CZ_DB_SETTINGS Table

SETTING_ID	SECTION_ NAME	DATA_ TYPE	Default VALUE	More information in...
DISPLAY_SUMMARY_ FULFILLMENT_ ACTION	UISERVER	String	True	DISPLAY_SUMMARY_ FULFILLMENT_ACTION on page 10-20

10.6.1.1 DISPLAY_SUMMARY_FULFILLMENT_ACTION

DISPLAY_SUMMARY_FULFILLMENT_ACTION determines whether the **Line Type** column appears in the Oracle Configurator Summary window during an update configuration session.

If it is a **new** configuration (that is, the end user is not updating an existing configuration), then the **Line Type** column does not appear, regardless of this setting's value.

If DISPLAY_SUMMARY_FULLMENT_ACTION is set to **True** or is not defined, then the **Line Type** column appears in the Summary window, if a reconfiguration.

If DISPLAY_SUMMARY_FULLMENT_ACTION is set to **False**, then the **Line Type** column does not appear in the Summary window.

For more information about updating configurations, see Section 11.1, "Modifying the TSO Functional Companions" on page 11-3.

10.6.2 CZ_CONFIG_ATTRIBUTES Table

You write a custom procedure and customize the downstream application so that it can use the configuration attribute data stored in CZ_CONFIG_ATTRIBUTES. For more information, see the *Oracle Configurator Methodologies* book.

10.6.3 CZ_CONFIG_EXT_ATTRIBUTES Table

The table CZ_CONFIG_EXT_ATTRIBUTES is the intermediate store of configuration attribute data between Oracle Configurator and Oracle Install Base. Oracle Configurator writes output data to the table when the IBAttribute Functional Companion runs. For more information about this Functional Companion, see Section 11.1.5.1, "Purpose of the IBAttribute Functional Companion" on page 11-21. Oracle Configurator compares the attribute values written to CZ_CONFIG_EXT_ATTRIBUTES with the values of the corresponding columns in the Install Base tables.

The layout and description of the CZ_CONFIG_EXT_ATTRIBUTES table appear in the Table 10–3, "CZ_CONFIG_EXT_ATTRIBUTES".

Table 10–3 CZ_CONFIG_EXT_ATTRIBUTES

Column Name	Null ?	PK ?	Type	Comments
CONFIG_HDR_ID	N	Y	NUMBER	Instance Header ID. Corresponds to CZ_CONFIG_ITEMS.INSTANCE_HDR_ID.
CONFIG_REV_NBR	N	Y	NUMBER	Instance Revision Number. Corresponds to CZ_CONFIG_ITEMS.INSTANCE_REV_NBR.
CONFIG_ITEM_ID	N	Y	NUMBER	Configuration Item ID. Corresponds to CZ_CONFIG_ITEMS.CONFIG_ITEM_ID.
ATTRIBUTE_LEVEL	N	Y	VARCHAR2(255)	Corresponds to CSI_I_EXTENDED_ATTRIBS.ATTRIBUTE_LEVEL.
ATTRIBUTE_NAME	N	Y	VARCHAR2(255)	Corresponds to CSI_I_EXTENDED_ATTRIBS.ATTRIBUTE_CODE.
ATTRIBUTE_GROUP	Y	N	VARCHAR2(255)	Corresponds to CSI_I_EXTENDED_ATTRIBS.ATTRIBUTE_GROUP.
ATTRIBUTE_VALUE	Y	N	VARCHAR2(255)	Corresponds to CSI_I_EXTENDED_ATTRIBS.ATTRIBUTE_VALUE.

The columns CONFIG_HDR_ID, CONFIG_REV_NBR, CONFIG_ITEM_ID, ATTRIBUTE_LEVEL, and ATTRIBUTE_NAME are the primary key for CZ_CONFIG_EXT_ATTRIBUTES.

For brevity, the preceding table does not include standard columns such as LAST_UPDATE_DATE.

10.6.4 Map Install Base Extended Attributes to CZ Attributes in Developer

You must map the following to each other for all trackable items:

- The extended attributes that you define in Install Base.
- The configuration attributes that you define on your configuration model in Configurator Developer.

The result of this mapping is that the values of configuration attributes produced during a session in Oracle Configurator remain associated with their trackable items in Install Base.

Module

Oracle Applications (Forms)

Responsibility

Oracle Installed Base Admin

Navigation

Setups > Extended Attribute Template

Steps

1. Define the desired extended attributes, associating them at the item group level. For information on defining Install Base attributes, see the chapter on setting up Install Base, and the *Oracle Install Base Implementation Guide*.
2. In Oracle Configurator Developer, modify your Model to add attribute Features and Properties.

   The methodology for modifying your Model is described in Oracle Configurator Methodologies, in the section on implementing configuration attributes for output. Consult that chapter for guidance, with these essential differences:

- Instead of descriptive flexfield segments and contexts, use Install Base extended attributes and groups.
- In place of the Property names ATTR_CONTEXT and ATTR_n_CONTEXT, use ATTR_GROUP and ATTR_n_GROUP.

Example

In Install Base, define an extended attribute at the Global access level with:

- Attribute Code = PORT_SPEED
- Attribute Name = Port Speed
- Attribute Category = EAST

In Configurator Developer, in a trackable BOM Model, define an attribute Feature with:

- Feature Name = Port_Speed
- Feature Options = 192 Kbps, 256 Kbps, 640 Kbps

For the Feature named Port_Speed, define a Property with:

- Property Name = ATTR_NAME
- Property Value = PORT_SPEED

For the trackable BOM Model, define a Property with:

- Property Name = ATTR_1_PATH
- Property Value = Port_Speed

For the trackable BOM Model, define a second Property with:

- Property Name = ATTR_1_MODE
- Property Value = 2

For the trackable BOM Model, define a third Property with:

- Property Name = ATTR_1_GROUP
- Property Value = EAST

10.6.5 MACD_FULFILLMENT_STRUCTURES Table

Oracle Telecommunications Service Ordering solution implementers need to create the MACD_FULFILLMENT_STRUCTURES table. This custom table is preserved during a schema upgrade or migration.

The table MACD_FULFILLMENT_STRUCTURES defines the types of changes relative to the installed baseline that can be made during a configuration session.

For information on how to use the MACD_FULFILLMENT_STRUCTURES table, see Section 11.1.3.3, "Setup for the Line Type Functional Companion" on page 11-10. Table 11–4 on page 11-11 describes how to populate this table.

The following example, Example 10–1, "Layout of MACD_FULFILLMENT_STRUCTURES" shows the layout of MACD_FULFILLMENT_STRUCTURES.

Example 10–1 Layout of MACD_FULFILLMENT_STRUCTURES

```
Name                           Null?    Type
------------------------------ -------- ----
ONT_TRANSACTION_TYPE_ID         NOT NULL NUMBER
CZ_DELTA_TYPE                   NOT NULL VARCHAR2(30)
INVENTORY_ITEM_ID               NOT NULL NUMBER
ORGANIZATION_ID                 NOT NULL NUMBER
PRIORITY                        NOT NULL NUMBER
```

The following example, Example 10–2, shows the SQL script that creates MACD_FULFILLMENT_STRUCTURES. This table does not contain seeded data. You must create your own data. Use the script in Example 10–2 to create this table. According to the comments in the script, you must create the table in the CZ schema, and insert data into the table using the CZ login (CZ/CZ) or the Apps login (APPS/APPS).

Example 10–2 Script to Create MACD_FULFILLMENT_STRUCTURES

```
-- Run as the APPS user to create this table in the CZ schema.
-- Insert data into this table using either the APPS/APPS or CZ/CZ login.
--
-- Data should be inserted as described in the Oracle Configurator
documentation.
-- A summary follows:
-- ONT_TRANSACTION_TYPE_ID is the ID created in the OE_TRANSACTION_TYPES_TL
table.
-- CZ_DELTA_TYPE is a change type that you specify (ADD/CHANGE/MOVE, etc.).
-- INVENTORY_ITEM_ID is the ID for the Inventory Item in the BOM.
-- ORGANIZATION_ID is the Organization ID for Item in the BOM.
-- PRIORITY (1, 2, 3, etc.) determines which CZ_DELTA_TYPE takes priority in a
session.

CREATE TABLE cz.macd_fulfillment_structures
(ont_transaction_type_id  NUMBER             NOT NULL,
```

```
cz_delta_type              VARCHAR2(30)       NOT NULL,
inventory_item_id          NUMBER             NOT NULL,
organization_id            NUMBER             NOT NULL,
priority                   NUMBER             NOT NULL);

-- Create a primary key to avoid duplicate records.
ALTER TABLE cz.macd_fulfillment_structures
ADD CONSTRAINT macd_fulfillment_structures_pk PRIMARY KEY
(ont_transaction_type_id, cz_delta_type, inventory_item_id,organization_
id,priority);

CREATE SYNONYM macd_fulfillment_structures FOR cz.macd_fulfillment_structures;
```

10.6.6 Mapping of CZ tables to Install Base Schema (CSI)

When adding trackable data, the system adds the item into the CSI_T_TXN_LINE_DETAILS table. If change occurs to a transaction value after a configuration update, then a comparison happens between CSI_T_TXN_LINE_DETAILS and CZ_CONFIG_ITEMS to report the change. The CSI_T_TRANSACTION_LINES table contains the transaction detail for the configured item. There is one transaction line for each source transaction. For more information, see:

- Section 3.4, "Configurator" on page 3-7
- Section 11.2, "Using the Oracle Configuration Interface Object" on page 11-25

Table 10–4 shows the mapping of the CZ tables to the Install Base schema.

Table 10–4 Mapping of CZ tables to IB (CSI) tables

CZ_TABLE.COLUMN	CSI_TABLE.COLUMN	Description
CZ_CONFIG_ITEMS.INSTANCE_HDR_ID	CSI_T_TXN_LINE_DETAILS.CONFIG_HDR_ID	Each configured component has its own unique instance header identification. The CONFIG_HDR_ID populates the INSTANCE_HDR_ID from the CZ_CONFIG_ITEMS table.
CZ_CONFIG_ITEMS.INSTANCE_REV_NBR	CSI_T_TXN_LINE_DETAILS.CONFIG_REV_NBR	Each configured component has its own unique instance revision number. The CONFIG_REV_NBR populates the INSTANCE_REV_NBR from the CZ_CONFIG_ITEMS table.
CZ_CONFIG_ITEMS.CONFIG_ITEM_ID	CSI_T_TXN_LINE_DETAILS.CONFIG_ITEM_ID	This is the unique identifier for the configured item.
CZ_CONFIG_ITEMS.TARGET_HDR_ID	CSI_T_II_RELATIONSHIPS.OBJ_CONFIG_INST_HDR_ID	This field is populated only for Connectors that point to the target.
CZ_CONFIG_ITEMS.TARGET_REV_NBR	CSI_T_II_RELATIONSHIPS.OBJ_CONFIG_REV_NBR	This field is populated only for Connectors that point to the target.
CZ_CONFIG_ITEMS.TARGET_ITEM_ID	CSI_T_II_RELATIONSHIPS.OBJ_CONFIG_ITEM_ID	This field is populated only for Connectors that point to the target.
CZ_CONFIG_ITEMS.CONFIG_ITEM_ID	CSI_T_TRANSACTION_LINES.SOURCE_TRANSACTION_ID	This is the ID of the item being configured.
CZ_CONFIG_ITEMS.UOM_CODE	CSI_T_TXN_LINE_DETAILS.UNIT_OF_MEASURE	This is the units of measurement code.
CZ_CONFIG_ITEMS.LOCATION_ID	CSI_T_TXN_LINE_DETAILS.LOCATION_ID	The Location ID for the item. For background information, see Section 11.1.2, "Specify the Default Location" on page 11-5.
CZ_CONFIG_ITEMS.LOCATION_TYPE_ID	CSI_T_TXN_LINE_DETAILS.LOCATION_TYPE_CODE	The Location Type Code for the item. For background information, see Section 11.1.2, "Specify the Default Location" on page 11-5.

Table 10–4 Mapping of CZ tables to IB (CSI) tables (Cont.)

CZ_TABLE.COLUMN	CSI_TABLE.COLUMN	Description
CZ_CONFIG_HDRS_V.BASELINE_REV_NBR	CSI_ITEM_INSTANCES.CONFIG_INST_REV_NUM CSI_T_TXN_LINE_DETAILS.CONFIG_INST_BASELINE_REV_NUM	CZ_CONFIG_HDRS_V.BASELINE_REV_NBR is a reference to CSI_ITEM_INSTANCES.CONFIG_INST_REV_NUM, which is the existing instance revision in Install Base that is being reconfigured. When Oracle Configurator writes transactions for a reconfiguration, it inserts this number in both CZ_CONFIG_HDRS_V.BASELINE_REV_NBR and CSI_T_TXN_LINE_DETAILS.CONFIG_INST_BASELINE_REV_NUM. For more information, see Section 2.1.3, "Reconfigure a Customer's Installed Services" and Section 11.2.1.1, "Identify Changes to a Configuration" on page 11-25.
CZ_CONFIG_EXT_ATTRIBUTES.ATTRIBUTE_NAME	CSI_T_EXTEND_ATTRIBS_V.ATTRIBUTE_CODE	The name of the extended attribute for the item. For more information, see Section 11.1.5, "Using Configuration Attributes with Install Base (IB)" on page 11-21.
CZ_CONFIG_EXT_ATTRIBUTES.ATTRIBUTE_VALUE	CSI_T_EXTEND_ATTRIBS_V.ATTRIBUTE_VALUE	The value of the extended attribute for the item. For more information, see Section 11.1.5, "Using Configuration Attributes with Install Base (IB)" on page 11-21.

10.7 Initialization Parameters

This section describes parameters that are specific to the TSO solution.

For information on the use of parameters in the initialization message, see the *Oracle Configurator Implementation Guide*.

When configuring instances of trackable components prior to their installation in Oracle Install Base, the system uses the following parameters:

- instance
- validation_context

For background on updating configuration instances, see Chapter 2, "TSO Business Process".

10.7.1 instance

The system uses the `instance` parameter when configuring instances of trackable components. This parameter lets you configure multiple instances in the same configuration session. In a host application, this corresponds to the end user's ability to select multiple components for configuration.

This is the only initialization parameter that can be specified multiple times in the initialization message.

In order to specify each instance that requires configuration, provide the Configuration Header ID (CZ_CONFIG_DETAILS_V.CONFIG_HDR_ID) as the value of the `header_id` sub-parameter. For example:

```
<instance header_id="17848"/>
<instance header_id="18333"/>
<instance header_id="23445"/>
```

10.7.2 validation_context

The system uses the `validation_context` parameter when configuring instances of trackable configuration components. The value sets the context in which the following occur:

- Validation of the current instances.
- Comparison of the current instances to the installed instances in Oracle Install Base.

Batch validation uses the `validation_context` parameter. For more information, see Section 10.8, "Batch Validation Parameters" on page 10-29.

The allowed values for the `validation_context` parameter appear in the following table, Table 10–5:

Table 10–5 Allowed Value for the validation_context Parameter

Value	Meaning
INSTALLED	Compare the revisions of the current instances only to the revisions of those instances that have been installed in Oracle Install Base.

Default The default value is INSTALLED. If you omit this parameter, the default value applies.

Not required.

10.8 Batch Validation Parameters

For background and details about batch validation, see the *Oracle Configurator Implementation Guide*.

TSO implementers must use one of the following optional parameters to the <batch_validate> element. Values are:

- validate_order

  The process should pass this value when validating orders, such as what Oracle Order Management does. This is the default value.

- validate_fulfillment

  The process should pass this value when validating fulfillment status, such as what Oracle Install Base does.

Syntax example:

```
<batch_validate validation_type="validate_order">
```

Usage example (Example 10–3):

Example 10–3 Example of Batch Validation Message

```
<batch_validate validation_type="validate_order">
  <initialize>
    <param name="context_org_id">204</param>
    <param name="config_creation_date">03-25-2001-19-30-02</param>
    <param name="calling_application_id">300</param>
    <param name="responsibility_id">20559</param>
    <param name="config_header_id">21361</param>
    <param name="config_rev_nbr">1</param>
```

```
    <param name="read_only">FALSE</param>
    <param name="save_config_behavior">new_revision</param>
    <param name="database_id">ap115sun_dev115</param>
  </initialize>
  <config_inputs>
    <option>
      <component_code>143-1490-1494</component_code>
      <quantity>1</quantity>
    </option>
    <option>
      <component_code>143-297</component_code>
      <quantity>1</quantity>
    </option>
  </config_inputs>
</batch_validate>
```

10.9 Programmatic Tools for TSO Development

You can use certain programmatic tools to develop a configuration model and deploy a runtime Oracle Configurator. For full background and details about these programmatic tools for development, see the *Oracle Configurator Implementation Guide*.

This section describes the tools or modified procedures for use with the TSO solution. For a list of these procedures, see Table 10–6, "Procedures Modified for TSO Development".

Table 10–6 Procedures Modified for TSO Development

API Name
CZ_CONFIG_API_PUB.COPY_CONFIGURATION on page 10-31
CZ_CONFIG_API_PUB.COPY_CONFIGURATION_AUTO on page 10-35
CZ_CF_API.VALIDATE on page 10-35

Table 10–7, " Custom Data Types for CZ_CONFIG_API_PUB" describes the custom data types that these procedures use. For background on custom data types, see the appropriate references in the *Oracle Configurator Implementation Guide*.

Table 10–7 Custom Data Types for CZ_CONFIG_API_PUB

Custom Type	Description
CZ_API_PUB.NUMBER_TBL_TYPE	Table of NUMBER.

10.9.1 COPY_CONFIGURATION

This procedure makes a copy of a saved configuration in the Oracle Configurator schema.

This procedure, COPY_CONFIGURATION in the CZ_CONFIG_API_PUB package, has been created for use with the TSO solution. It is a variation of COPY_CONFIGURATION in the CZ_CF_API package. For a description of the CZ_CF_API package, see the *Oracle Configurator Implementation Guide*. The most important difference is that this TSO procedure has two additional OUT parameters: `x_orig_item_id_tbl` and `x_new_item_id_tbl`. These parameters inform host applications about changes to the Configuration Item IDs (CZ_CONFIG_ITEMS.CONFIG_ITEM_ID) in the configuration in a case where you are copying component instances that are not yet installed in Oracle Install Base. These parameters are of type CZ_API_PUB.NUMBER_TBL_TYPE, which is a table of the NUMBER type. As used by the COPY_CONFIGURATION procedure, these output tables are indexed from 1 to *n*, where *n* is the number of Configuration Item IDs changed from the source configuration to the new configuration. You can use the index number of the IDs stored in the tables to map the Configuration Item IDs in the original configuration to the Configuration Item IDs in the new copy of that configuration. Example 10–4, "Mapping Original Configuration Item IDs to News IDs When Copying a Configuration" shows how a host application might process these outputs using Oracle Configurator.

Example 10–4 Mapping Original Configuration Item IDs to News IDs When Copying a Configuration

```
...
FORALL i IN x_orig_item_id_tbl.FIRST..x_orig_item_id_tbl.LAST
  UPDATE my_order_lines
  SET config_item_id = x_new_item_id_tbl(i)
  WHERE my_order_header = l_header
    AND config_hdr_id = x_config_hdr_id
    AND config_rev_nbr = x_config_rev_nbr
    AND config_item_id = x_orig_item_id_tbl(i);
...
```

The non-TSO version of COPY_CONFIGURATION, in the package CZ_CF_API, has been modified to create an exception if it generates any new Configuration Item IDs. This averts data corruption in Order Management, Order Capture, and Contracts. If you encounter this exception, you must apply the appropriate patch for your host application that invokes the new COPY_CONFIGURATION procedure from CZ_CONFIG_API_PUB.

For background on configurations, see the chapter on managing configurations in the *Oracle Configurator Implementation Guide*.

10.9.1.1 Considerations Before Running

Prerequisites

The configuration that you are copying must exist. If the configuration has been logically deleted, then you can pass **0** as the value of `p_handle_deleted_flag` to enable copying.

Timing

You should use this procedure every time you copy a configuration. The procedure ensures copying of all inputs, outputs, attributes, and messages.

Warnings

If the configuration does not exist, or if the copy fails, `x_msg_count` will be greater than zero, and `x_msg_data` will contain error information. Check `x_return_status` for the return status.

10.9.1.2 Syntax and Parameters

The syntax for this procedure is:

```
PROCEDURE copy_configuration(p_api_version          IN  NUMBER
                            ,p_config_hdr_id        IN  NUMBER
                            ,p_config_rev_nbr       IN  NUMBER
                            ,p_copy_mode            IN  VARCHAR2
                            ,x_config_hdr_id        OUT NOCOPY  NUMBER
                            ,x_config_rev_nbr       OUT NOCOPY  NUMBER
                            ,x_orig_item_id_tbl     OUT NOCOPY  CZ_API_PUB.number_tbl_type
                            ,x_new_item_id_tbl      OUT NOCOPY  CZ_API_PUB.number_tbl_type
                            ,x_return_status        OUT NOCOPY  VARCHAR2
                            ,x_msg_count            OUT NOCOPY  NUMBER
                            ,x_msg_data             OUT NOCOPY  VARCHAR2
                            ,p_handle_deleted_flag  IN  VARCHAR2 := NULL
```

```
,p_new_name              IN  VARCHAR2 := NULL
);
```

Table 10–8 on page 10-33describes the parameters for the COPY_ CONFIGURATION procedure.

Table 10–8 Parameters for the COPY_CONFIGURATION Procedure

Parameter	Data Type	Mode	Note
p_api_version	number	in	Required. See the description of API Version Numbers in the *Oracle Configurator Implementation Guide*.
p_config_hdr_id	number	in	Required. Specifies the Configuration Header ID of the source configuration to copy.
p_config_rev_nbr	number	in	Required. Specifies the Configuration Revision Number of the source configuration to copy.
p_copy_mode	varchar2	in	Required. Flag that specifies whether creating a new Configuration Header ID or new Configuration Revision Number has one of the following values: ■ CZ_API_PUB.G_NEW_HEADER_COPY_MODE, which creates a new Configuration Header ID. Used in the case of reordering. Equivalent to the value `new_config` for the initialization parameter `save_config_behavior`. ■ CZ_API_PUB.G_NEW_REVISION_COPY_MODE, which creates a new Configuration Revision Number. Used in the case of reconfiguring or repricing. Equivalent to the value `new_revision` for the initialization parameter `save_config_behavior`.
x_config_hdr_id	number	out	Identifies the Configuration Header ID of the new copy of the source configuration.
x_config_rev_nbr	number	out	Identifies the Configuration Revision Number of the new copy of the source configuration.

Table 10–8 Parameters for the COPY_CONFIGURATION Procedure (Cont.)

Parameter	Data Type	Mode	Note
x_orig_item_id_tbl	CZ_API_ PUB.number_tbl_ type	out	Indexed table of Configuration Item IDs from the source configuration. Empty if the configuration does not contain trackable instances.
x_new_item_id_tbl	CZ_API_ PUB.number_tbl_ type	out	Indexed table of Configuration Item IDs from the new copy of the source configuration. Empty if the configuration does not contain trackable instances.
x_return_status	varchar2	out	Standard out parameter. Value is: FND_ API.G_RET_STS_SUCCESS, FND_API.G_ RET_STS_ERROR, or FND_API.G_RET_ STS_UNEXP_ERROR.
x_msg_count	number	out	Standard out parameter.
x_msg_data	varchar2	out	Standard out parameter.
p_handle_deleted_ flag	varchar2	in	Specifies how to handle an attempt to copy a logically deleted source configuration. If the source configuration is logically deleted, and the value passed to this parameter is **0**, then the source configuration is undeleted so that the copy can proceed. If the value passed is null, then an error is raised by the attempt to copy.
p_new_name	varchar2	in	The new name applied to the copied configuration.

10.9.1.3 Considerations After Running

Results

This procedure copies all database records associated with a configuration to a new Configuration Header ID and Configuration Revision Number. The changes that the procedure made are not automatically committed. When you call this procedure, you must check `x_return_status` and commit the transaction if the copy was successful.

Troubleshooting

Examine x_return_status. If it does not indicate success, then examine x_msg_count for the number of error messages, and x_msg_data for the error messages themselves.

10.9.2 COPY_CONFIGURATION_AUTO

This procedure runs COPY_CONFIGURATION within an autonomous transaction. If the copy is successful, new data will be committed to the database without impacting the caller's transaction.

For more information, see "COPY_CONFIGURATION" on page 10-31.

10.9.3 VALIDATE

For TSO, you must call this procedure with the parameter p_validation_type, which Table 10–9 on page 10-36 describes.

When Oracle Configurator calls this procedure during batch validation, it uses the following values:

- CZ_API_PUB.VALIDATE_ORDER

  This value should be passed when validating orders, such as what Oracle Order Management does. This is the default value.

- CZ_API_PUB.VALIDATE_FULFILLMENT

  This value should be passed when validating fulfillment status, such as what Oracle Install Base does.

For other essential information about this procedure, see the *Oracle Configurator Implementation Guide*.

10.9.3.1 Syntax and Parameters

The syntax for this procedure is:

```
PROCEDURE VALIDATE ( config_input_list IN CFG_INPUT_LIST,
                     init_message IN VARCHAR2,
                     config_messages IN OUT CFG_OUTPUT_PIECES,
                     validation_status IN OUT NUMBER,
                     URL IN VARCHAR2 DEFAULT FND_PROFILE.Value('CZ_UIMGR_URL'),
                     p_validation_type IN VARCHAR2 DEFAULT CZ_API_PUB.VALIDATE_
ORDER));
```

Table 10–9 on page 10-36 describes the parameter for the VALIDATE procedure with special meaning for the TSO solution. For information on the other parameters for the VALIDATE procedure, see the *Oracle Configurator Implementation Guide*.

Table 10–9 Special Parameter for the VALIDATE Procedure

Parameter	Data Type	Mode	Note
p_validation_type	varchar2	in	The possible values are CZ_API_PUB.VALIDATE_ORDER, CZ_API_PUB.VALIDATE_FULFILLMENT, and CZ_API_PUB.INTERACTIVE. The default is CZ_API_PUB.VALIDATE_ORDER.

11

Set Up Configurator Functional Companions

The main topics in this chapter are:

- Section 11.1, "Modifying the TSO Functional Companions" on page 11-3
- Section 11.2, "Using the Oracle Configuration Interface Object" on page 11-25

Reference Material

This chapter describes setup tasks that you must perform to Oracle Configurator that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on functionality, setting up, or using Oracle Configurator, refer to:

- Section 3.4, "Configurator" on page 3-7
- *Oracle Configurator Developer User's Guide*
- *Oracle Configurator Implementation Guide*
- *Oracle Configurator Installation Guide*
- *Oracle Configuration Interface Object (CIO) Developer's Guide*
- *Oracle Configurator Methodologies.*
- *Oracle Configurator Release Notes*

Before You Begin

This chapter explains:

- The role of Oracle Configurator Functional Companions in a Telecommunications Service Ordering solution, and how to modify these

Functional Companions where necessary for your requirements. The Functional Companions provide a certain amount of the TSO functionality without modification by installing them. Other optional functionality requires you to modify the Functional Companions. For more information, see Section 11.1.1, "Overview" on page 11-3.

- How to use certain features of the Oracle Configuration Interface Object (CIO) that are specifically designed to support a Telecommunications Service Ordering solution. For more information, see Section 11.2, "Using the Oracle Configuration Interface Object" on page 11-25.

In addition to this chapter, there is another chapter on setting up Configurator for use with the Telecommunications Services Ordering solution, Chapter 10, "Set Up Configurator and Customize the Solution".

Before you can set up Oracle Configurator for use with the Oracle Telecommunications Service Ordering solution, you must set up Oracle Bill of Materials, Oracle Install Base, and Oracle Order Management. For more information, see:

- Chapter 7, "Set Up Bill of Materials"
- Chapter 8, "Set Up Install Base"
- Chapter 5, "Set Up Order Management"

After You Set Up Oracle Configurator

After you set up Oracle Configurator, you can set up Oracle Quoting. Note, however, that Quoting is also dependent on the setup of Oracle Advanced Pricing. For more information, see Chapter 12, "Set Up Quoting".

11.1 Modifying the TSO Functional Companions

This section describes the use of several defined Functional Companions used in updating installed configurations of Container Models.

Main topics in this section include:

- Section 11.1.1, "Overview" on page 11-3
- Section 11.1.2, "Specify the Default Location" on page 11-5
- Section 11.1.3, "Specify the Line Type" on page 11-9
- Section 11.1.4, "Change the Instance Name" on page 11-18
- Section 11.1.5, "Using Configuration Attributes with Install Base (IB)" on page 11-21

11.1.1 Overview

Updating installed configurations of Container Models relies on the use of one or more Functional Companions, depending on how much you want to customize the Telecommunications Services Ordering solution, if at all. The available solutions are the:

- Default solution, which requires no or minimal implementation work on your part.
- Customized solution, which requires customization to achieve your desired results.

Table 11–1 on page 11-3 lists the Functional Companions that Telecommunications Services Ordering uses and indicates the solution for which each is applicable.

Table 11–1 Functional Companions for Telecommunications Services Ordering

Solution	Name	Description	See...
Default	Default Location	Sets the default Location of configured components	Section 11.1.2, "Specify the Default Location" on page 11-5

Table 11–1 (Cont.) Functional Companions for Telecommunications Services Ordering (Cont.)

Solution	Name	Description	See...
Customized	Line Type[1]	Sets the Line Type of the configured component	Section 11.1.3, "Specify the Line Type" on page 11-9
Customized	IBAttribute	Collects configuration attributes of a configured component	Section 11.1.5.3, "Setup for the IBAttribute Functional Companion" on page 11-22
Customized	Instance Name	Sets the name of an instance of a configured component	Section 11.1.4, "Change the Instance Name" on page 11-18

[1] Requires a patch for use with Oracle Order Management. See Section 11.1.3, "Specify the Line Type" on page 11-9.

Note: The ability to update installed configurations of Container Models is designed for and intended for use by the telecommunications service industry. Users in this area of business need to configure and update a large network of connected components, such as a long-distance telephone network. Installed configurations such as these contain only serviceable items; they do not contain manufacturable items (such as items based on ATO BOM Models) or items that can be shipped. Therefore, this functionality is not intended for configuration models that are manufacturable or contain any manufacturable sub-components.

For more information about updating installed configurations of Container Models, see Chapter 2, "TSO Business Process".

You should consult these other documents for information on the tasks in this section:

- For information on how to build a configuration model that enables you to update installed configurations of Container Models, see Chapter 2, "TSO Business Process".
- For information on how to write and compile Functional Companions, and on how to incorporate them into your configuration model, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.
- For information on how to install Functional Companions, see the *Oracle Configurator Installation Guide*.

11.1.2 Specify the Default Location

The Default Location Functional Companion has been provided to enable you to test the TSO flow. To query and select location information from the TCA (or another source), a different Functional Companion must be written.

The Default Location Functional Companion is used for implementing the updating of installed configurations of Container Models.

11.1.2.1 Purpose of the Default Location Functional Companion

The Default Location Functional Companion determines and sets a default Location ID and Location Type Code on a trackable component instance and all its trackable descendants. Having these default values assigned allows you to test the TSO flow.

To understand the need for the Location ID and Location Type Code, see Background for Locations.

Note: If you run the Custom Location Functional Companion after running the Default Location Functional Companion, the default Locations are overwritten with the customized Locations.

Background for Locations

Oracle Sales Online lets you create multiple addresses for customers. Each address is identified by a Location ID, which the Oracle Sales Online end user cannot see. At the other end of the quote-to-installation flow, Oracle Install Base requires that every installed component instance have a Location ID, to identify the location at which an item has been installed. Along the course of this flow, the Location Type Code points to the database table containing the detailed information for the address that the Location ID identifies.

An Oracle Configurator configuration model normally represents only the configurable aspects of a component, and thus does not have any innate means of expressing the location of a configured component. Consequently, there is no built-in equivalent to the Location ID. For a trackable component to be configured and later accepted by Install Base, a Location ID must be assigned to every configured component.

11.1.2.2 Example Code for the Default Location Functional Companion

For the full source code of the Default Location Functional Companion, see Example B–1, "Setting the Default Location (DefaultLocation.java)" on page B-3.

11.1.2.3 Setup for the Default Location Functional Companion

This section describes what you must do before using the Default Location Functional Companion.

1. In Oracle Install Base, you have performed the required tasks for setting up locations and Location IDs.
2. In Oracle Sales Online, you have defined your Customers.

   For more information, see Background for Locations on page 11-5.
3. A compiled version of the Default Location Functional Companion is installed as part of the Oracle Applications archive file, `apps.zip`. You do *not* have to recompile or install this Functional Companion. For information on installing Functional Companions, see the *Oracle Configurator Installation Guide*.
4. To modify the Default Location Functional Companion, see Modify the Default Location Functional Companion on page 11-7.
   - You can modify the Default Location Functional Companion to change the default Location ID and Location Type Code that is assigned to trackable components.
   - If you do not modify the Default Location Functional Companion, then it sets a default Location ID and Location Type Code on the configured trackable component and all its trackable descendants. The default Location ID is set to the first record in the table HZ_PARTY_SITES.
   - If there is at not least one Location ID record (in the table HZ_PARTY_SITES), then the Functional Companion generates a RuntimeException. This might be the case if no Customer data has been defined.
5. Define a Functional Companion rule that associates the Default Location Functional Companion with your Model, setting the **Definition** of the Functional Companion rule as the following Table 11–2 shows.

Table 11–2 Rule Definition Attributes and Values

Rule Definition Attribute	Definition Value
Base Component	The highest node on which you want to set the Location ID. The Default Location Functional Companion sets a Location ID on this component and all of its trackable descendants.
Type	Event-Driven
Implementation	Java

Table 11–2 Rule Definition Attributes and Values (Cont.)

Rule Definition Attribute	Definition Value
Program String	The name of the Java class that implements the Functional Companion. For the example, this is `DefaultLocation` (Example B–1 on page B-3).

6. Perform the usual procedure for testing a Functional Companion. This includes generating the Active Model and using the Test/Debug module.

11.1.2.4 Effects of the Default Location Functional Companion

The Default Location Functional Companion extends the CIO class `AutoFunctionalCompanion`, and implements its method `onSave()`. This means that the code is executed when the current configuration is saved, such as when the user clicks the **Done** button at the end of a configuration session.

The `onSave()` method first checks whether the associated component is a trackable component, using `IRuntimeNode.isTrackableRoot()`. If it is, then the private method `initializeDefaultLocation()` executes a database query that returns a value to use for the default Location ID. The private method `setLocation()` sets that value on the trackable component being configured, and on all its trackable descendants. `setLocation()` employs `IRuntimeNode.getSummaryChildren()` to iterate through all the trackable descendants. The method `getSummaryChildren()` gets all of the trackable children of the configured components, including those that are discontinued and those that are network link components.

In programming terms, a network link is defined as a BOM Model having exactly two children, both of which are Connectors. Discontinued child components are those having a Line Type of DISCONTINUED. For background information, see Section 11.1.3.3, "Setup for the Line Type Functional Companion" on page 11-10. Discontinued child components appear on the Summary window, but are not saved as part of the configuration of the component.

11.1.2.5 Modify the Default Location Functional Companion

For information on using the classes and methods of the Configuration Interface Object (CIO), see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

The full source code appears in Example B–1, "Setting the Default Location (DefaultLocation.java)" on page B-3. If you modify this source code, then you need to recompile and install the Functional Companion as described in the *Oracle*

Configuration Interface Object (CIO) Developer's Guide and the *Oracle Configurator Installation Guide*.

> **Note:** If you customize this Functional Companion, you must place it in a Java package of your own, and not in `Oracle.*`.

You may want to change the database query that returns the value of the default Location ID. The following code fragment from Example B–1, "Setting the Default Location (DefaultLocation.java)" shows this query:

```
...
 String fetch = "SELECT party_site_id " +
        "FROM apps.hz_party_sites " +
        "WHERE sysdate BETWEEN NVL(start_date_active, sysdate) and " +
        "NVL(end_date_active, sysdate) " +
        "AND ROWNUM < 2 " ;
...
```

When reformatted and run in SQL*Plus, this query might return a value such as the following (depending on the contents of the database instance):

```
PARTY_SITE_ID
-------------
       1000
```

To obtain Location information from the table HZ_LOCATIONS instead of HZ_PARTY_SITES, change the appropriate references in the lines of Example B–1, "Setting the Default Location (DefaultLocation.java)" that appear below:

```
...
static String m_defaultLocationTypeCode = "HZ_PARTY_SITES";
...
"FROM apps.hz_party_sites " +
...
throw new RuntimeException("Unable to initialize a Default Location from
APPS.HZ_PARTY_SITES table.");
...
```

11.1.3 Specify the Line Type

The Line Type Functional Companion is recommended, but not required for implementing the updating of installed configurations of Container Models.

If you do not install the Line Type Functional Companion, then your fulfilled items will lack Line Types (also known as Actions). In Service Fulfillment Manager (SFM), the Line Types for the items will be null.

If you install the Line Type Functional Companion without modification, and complete the setup steps described in this chapter, then your fulfilled orders will have Line Types in SFM. The Line Types will be those defined in the Line Type Functional Companion and the associated setup.

If you install the Line Type Functional Companion, modify it, and complete the setup steps described in this chapter, then your fulfilled orders will have Line Types in SFM that meet your specific requirements.

If you install the Line Type Functional Companion, with or without modification, and do not install the prerequisite patch for Oracle Order Management, then your items will record only a single Line Type, ADD, regardless of the Line Type that was set by the Line Type Functional Companion.

Note: For Oracle Order Management to support multiple Line Types in a configuration, you must install an essential prerequisite patch. See the *Oracle Configurator Release Notes* for more information.

11.1.3.1 Purpose of the Line Type Functional Companion

The Line Type Functional Companion determines and sets the Line Type value string for the associated node and its trackable descendants by detecting the type of change that was made to the component being configured or reconfigured, relative to the latest valid baseline recorded in Oracle Install Base.

The Line Type is a user-defined string that indicates the fulfillment actions to be applied to the configured component instances. The Functional Companion assigns the Line Type during a configuration session when the configuration is saved or when the Summary window appears.

11.1.3.2 Example Code for the Line Type Functional Companion

For the full source code of the Line Type Functional Companion, see Example B–2, "Setting the Line Type (LineTypeFunc.java)" on page B-5.

11.1.3.3 Setup for the Line Type Functional Companion

This section describes what you must do before using the Line Type Functional Companion.

1. After optionally modifying it, compile and install the Line Type Functional Companion. The full source code of the Line Type Functional Companion appears in Example B–2, "Setting the Line Type (LineTypeFunc.java)" on page B-5.
2. To display the Line Types in the Summary window of the runtime Oracle Configurator, ensure that DISPLAY_SUMMARY_FULFILLMENT_ACTION is set correctly in the CZ_DB_SETTINGS table, as described in Section 10.6.1, "TSO Solution CZ_DB_SETTINGS Parameters" on page 10-20.
3. In Oracle Order Management, define the desired Line Types, by defining Transaction Types that have a Transaction Type Code of Line.

   These types are stored in the Order Management table ONT.OE_TRANSACTION_TYPES_TL. The name you enter in the Transaction Types window is stored as NAME. Oracle Applications stores a numeric code for the type, TRANSACTION_TYPE_ID. The NAME value is used to display the Line Type values on the Summary window of the runtime Oracle Configurator. The following table, Line Types Defined in OE_TRANSACTION_TYPES_TL, shows a set of Line Types that you might define. The IDs shown are example values; the actual values of the IDs depend on the particular database instance.

   These types are passed downstream to SFM to indicate the type of fulfillment action to perform.

Table 11–3 Line Types Defined in OE_TRANSACTION_TYPES_TL

TRANSACTION_TYPE_ID	NAME
2516	MOVE
2476	ADD
2477	CHANGE
2556	DISCONTINUED

Use the following query to see the other Transaction Types currently defined:

```
SELECT transaction_type_id, name, description FROM ont.oe_transaction_types_
tl;
```

4. Create a new table, MACD_FULFILLMENT_STRUCTURES.

   For a description of the table, see Section 10.6.5, "MACD_FULFILLMENT_STRUCTURES Table" on page 10-23.

   The table MACD_FULFILLMENT_STRUCTURES is **not** seeded in the Oracle Applications database. Example 10–1 on page 10-24 shows the layout of this table, and Example 10–2 on page 10-24 shows a script that creates this table.

   This table defines the types of changes relative to the installed baseline that can happen during a configuration session. It also defines the correspondences between the Line Types in downstream applications and the change (delta) types that can happen in the runtime Oracle Configurator relative to the installed baseline. Specifically, the table contains mappings between NAME in the Order Management table (ONT.OE_TRANSACTION_TYPES_TL) and CZ_DELTA_TYPE in the custom table (MACD_FULFILLMENT_STRUCTURES), based on common values for their associated transaction type IDs. Table 11–5, "Mapping of Transaction Type IDs" on page 11-13 shows this mapping.

   As an alternative to using the MACD_FULFILLMENT_STRUCTURES table, you can also encode the mappings in your own version of the Line Type Functional Companion. This alternative is not described here.

5. In the table MACD_FULFILLMENT_STRUCTURES, insert rows that express a set of the types of changes that can happen in the runtime Oracle Configurator relative to the installed baseline. For guidance, see Table 11–4, "Values To Enter in MACD_FULFILLMENT_STRUCTURES" on page 11-11.

Table 11–4 Values To Enter in MACD_FULFILLMENT_STRUCTURES

Column	Value
ONT_TRANSACTION_TYPE_ID	Enter a value that corresponds to a value of ONT.OE_TRANSACTION_TYPES_TL.TRANSACTION_TYPE_ID. For the possible values of TRANSACTION_TYPE_ID, see Table 11–3, "Line Types Defined in OE_TRANSACTION_TYPES_TL" on page 11-10. Your values for ONT_TRANSACTION_TYPE_ID define the correspondences between Order Management Line Types and OC change types. Note that this correspondence is not necessarily one-to-one. Several OC change types might all map to the same Line Type in Order Management.

Table 11–4 Values To Enter in MACD_FULFILLMENT_STRUCTURES (Cont.)

Column	Value
CZ_DELTA_TYPE	Enter the change type. The change types that work with the Line Type Functional Companion appear in the column CZ_DELTA_TYPE in Table 11–6, "Example Baseline Change Types in MACD_FULFILLMENT_STRUCTURES" on page 11-13. Each change type corresponds to one of a set of methods in the CIO interface `IRuntimeNode` that detects all the types of baseline changes that are meaningful for Oracle Configurator. These methods appear in Table 11–8, "Methods in IRuntimeNode for Detecting Baseline Changes" on page 11-15.
PRIORITY	Enter the desired priority for this change type. This priority is the importance of this change type relative to the other change types that CZ_DELTA_TYPE defined.
ORGANIZATION_ID	The Organization ID for the configurable component. The combination of this value with INVENTORY_ITEM_ID identifies the individual component.
INVENTORY_ITEM_ID	The Inventory Item ID for the configurable component. The combination of this value with ORGANIZATION_ID identifies the individual component.

Table 11–5, "Mapping of Transaction Type IDs" on page 11-13 shows an example of the mapping between NAME in the Order Management table OE_TRANSACTION_TYPES_TL and CZ_DELTA_TYPE in the custom table MACD_FULFILLMENT_STRUCTURES, which is based on common values for their associated transaction type IDs. This mapping represents the relationship that you are defining between change types and Line Types. The IDs in the table are examples of values; the actual values of the IDs depend on the particular database instance.

Table 11–5 Mapping of Transaction Type IDs

MACD_FULFILLMENT_STRUCTURES Columns		OE_TRANSACTION_TYPES_TL Columns	
CZ_DELTA_TYPE	ONT_ TRANSACTION_ TYPE_ID	TRANSACTION _TYPE_ID	NAME
DELETE	2556	2556	DISCONTINUED
ADD	2476	2476	ADD
VALUE_CHANGE	2477	2477	CHANGE
TARGET_CHANGE	2516	2516	MOVE
ATTRIBUTE_ CHANGE	2477	2477	CHANGE
LOCATION_CHANGE	2516	2516	MOVE
NAME_CHANGE	2477	2477	CHANGE
CHILD_CHANGE	2477	2477	CHANGE

6. Repeat the insertion of a complete set of OC change types for each of your configurable components that Install Base tracks.

   For the Line Type Functional Companion to work properly, you must insert a record for all of the OC change types (DELETE through CHILD_CHANGE) for each component, even if you are only interested in a few of these change types.

   Table 11–6, "Example Baseline Change Types in MACD_FULFILLMENT_ STRUCTURES" on page 11-13 shows an example of a set of OC change type records that you might define for a component with ORGANIZATION_ID 204 and INVENTORY_ITEM_ID 6920.

Table 11–6 Example Baseline Change Types in MACD_FULFILLMENT_STRUCTURES

ONT_ TRANSACTION _TYPE_ID	CZ_DELTA_TYPE	PRIORI TY	ORGANIZA TION_ID	INVENTORY_ ITEM_ID
2556	DELETE	1	204	6920
2476	ADD	2	204	6920
2477	VALUE_CHANGE	3	204	6920
2516	TARGET_CHANGE	4	204	6920

Table 11–6 Example Baseline Change Types in MACD_FULFILLMENT_STRUCTURES

ONT_ TRANSACTION _TYPE_ID	CZ_DELTA_TYPE	PRIORITY	ORGANIZATION_ID	INVENTORY_ ITEM_ID
2477	ATTRIBUTE_ CHANGE	5	204	6920
2516	LOCATION_CHANGE	6	204	6920
2477	NAME_CHANGE	7	204	6920
2477	CHILD_CHANGE	8	204	6920

7. Define a Functional Companion rule that associates the Line Type Functional Companion with your Model, setting the **Definition** of the Functional Companion rule as Table 11–7 shows.

Table 11–7 Rule Definition Attribute and Definition Value

Rule Definition Attribute	Definition Value
Base Component	The component whose instances need their Line Types updated. The update applies to all trackable descendants of the component.
Type	Event-Driven
Implementation	Java
Program String	The name of the Java class that implements the Functional Companion. For the example, this is `LineTypeFunc` (Example B–2 on page B-5).

8. Perform the usual procedure for testing a Functional Companion. This includes generating the Active Model and using the Test/Debug module.

   The Line Type Functional Companion uses AppsContext for connecting to the database. This requires a login that generates an ICX session ticket. Therefore, your initialization message must pass the parameters `database_id` (instead of `alt_database_name`) and `icx_session_ticket` (instead of `gwyuid`, `user`, and `pwd`). To use the **Test** button in Configurator Developer, you must log in with an Oracle Applications ID and password.

11.1.3.4 Effects of the Line Type Functional Companion

The Line Type Functional Companion:

- Extends the Oracle Configuration Interface Object (CIO) class `AutoFunctionalCompanion`
- Implements its methods `onSummary()` and `onSave()`.

This means that the code executes when the Summary window appears and displays the Line Type values. The code also executes when the current configuration is saved, such as when the user clicks the **Done** button at the end of a configuration session.

The Functional Companion begins by setting up variables that mirror the baseline change types that are recorded in MACD_FULFILLMENT_STRUCTURES:

```
public static String ADD  = "ADD" ;
public static String QUANTITY = "VALUE_CHANGE";
public static String DELETE =  "DELETE";
public static String TARGET_CHANGE = "TARGET_CHANGE";
public static String ATTRIBUTE_CHANGE = "ATTRIBUTE_CHANGE";
public static String LOCATION_CHANGE = "LOCATION_CHANGE";
public static String NAME_CHANGE = "NAME_CHANGE";
public static String CHILD_CHANGE = "CHILD_CHANGE";
```

Table 11–8, "Methods in IRuntimeNode for Detecting Baseline Changes" lists the methods in the CIO interface `IRuntimeNode` that correspond to the change types shown in Table 11–6 on page 11-13 and how these relate to the Line Types shown in Table 11–3 on page 11-10. For a description of these methods, see Table 11–11 on page 11-26.

Table 11–8 Methods in IRuntimeNode for Detecting Baseline Changes

Method	Change Type	Line Type
`isDeleteChanged()`	DELETE	DISCONTINUED
`isAddChanged()`	ADD	ADD
`isValueChanged()`	VALUE_CHANGE	CHANGE
`isTargetChanged()`	TARGET_CHANGE	MOVE
`isAttributeChanged()`	ATTRIBUTE_CHANGE	CHANGE
`isLocationChanged()`	LOCATION_CHANGE	MOVE
`isNameChanged()`	NAME_CHANGE	CHANGE
`isChildChanged()`	CHILD_CHANGE	CHANGE

When the end user requests a Summary or Save in the configuration session, the implemented body of `onSummary()` begins:

```
node = getRuntimeNode()
...
initializeDataForInstance(node)
...
```

The Functional Companion executes a query into a result set. This query, for the configured component (identified by INVENTORY_ITEM_ID and ORGANIZATION_ID), retrieves all the matching records in MACD_FULFILLMENT_STRUCTURES (all the possible types of changes).

Then, put each entry in this result set into a slot in an array of `czFulfillmentRecords` (which now define all the possible types of changes for the configured node).

Now, using the array of change types, set the action on the component:

```
setFulfillmentAction(node)
```

If any change happened to the node:

```
if (node.isChanged()
```

Set the Line Type, which invokes determineFulfillmentAction():

```
node.setLineType(new Integer(determineFulfillmentAction(node)))
```

Then determine the appropriate action for the node:

```
determineFulfillmentAction(node)
```

First, set variables for the possible RuntimeNode **change** methods:

```
boolean add = node.isAddChanged();
boolean delete = node.isDeleteChanged();
// and so on
```

Then check whether the node is a network link (a BOM Model having exactly two children, both of which are Connectors).

```
boolean networkLink = isNetworkLinkAndLocationChanged(node);
```

Then iterate through the CzFullfillmentRecord's returned from the query to the MACD_FULFILLMENT_STRUCTURES table.

Get the next fulfilment record:

```
CzFullfillmentRecord czRecord = (CzFullfillmentRecord)itr.next();
```

Then get the ID (ONT_TRANSACTION_TYPE_ID) and the delta string (CZ_DELTA_TYPE) for it:

```
int fulfillmentAction =  czRecord.getFulfillmentActionId();
String delta = czRecord.getCZDelta();
```

As a special case, first verify a target_change on a network link, which is defined as a Line Type of MOVE.

```
if (networkLink && delta.equalsIgnoreCase(TARGET_CHANGE) ) {
```

Then check each of your other defined change types to see if that change happened to the current node, and if your previously defined string (such as ADD = "ADD") matches a CZ_DELTA_TYPE value ("ADD"):

```
if (delta.equalsIgnoreCase(ADD) && add) {
```

If the type part of the record matches ADD, and the change that happened (as detected by `node.isAddChanged()`) is what was reported by the RuntimeNode, then return this is the change to set as the Line Type, so return the ID for the change (ONT_TRANSACTION_TYPE_ID) using `getFulfillmentActionId()`:

```
return fulfillmentAction;
```

Repeat the test for each change type.

Since the `CzFulfillmentRecord`s were ordered by the PRIORITY value from MACD_FULFILLMENT_STRUCTURES (**order by PRIORITY**), the order of tests must match that priority, so that the highest-priority change is caught first, and the function returns right there. So `node.setLineType()` sets the type returned by the tests.

Then you recurse over the child nodes of the component associated with the Functional Companion, to set the fulfillment action (Line Type) on each of them too:

```
for (Iterator itr = node.getSummaryChildren().iterator(); itr.hasNext(); ) {
 setFulfillmentAction((RuntimeNode)itr.next());
}
```

... which involves determining the correct action for the node, running through the bank of tests, as with the first node.

During the configuration session, the end user can make several different changes of the types that are detectable by the Functional Companion's set of methods, so

the definition of PRIORITY in MACD_FULFILLMENT_STRUCTURES indicates which of these changes passes to SFM.

11.1.3.5 Modify the Line Type Functional Companion

If you modify your Model and the database tables in accordance with the steps in Section 11.1.3.3, "Setup for the Line Type Functional Companion" on page 11-10, then you do not need to modify the Line Type Functional Companion. If you define different Line Type values, then you need to modify the set of test variables (`ADD` and `add`, and so on) and the `isChanged()` methods that correspond to them.

11.1.4 Change the Instance Name

The Instance Name Functional Companion is recommended, but not required for implementing the updating of installed configurations of Container Models.

11.1.4.1 Purpose of the Instance Name Functional Companion

The Instance Name Functional Companion enables an end user of the runtime Oracle Configurator to change the name of a component instance so that the new name is retained across configuration sessions, and appears if the configuration containing the component is restored.

When a component that can be instantiated multiple times appears in the user interface of the runtime Oracle Configurator, its name includes a distinguishing number.

It can be very helpful to the end user to be able to change the name of a runtime component instance, to distinguish it more easily from other instances with names that are similar, although they are never identical.

While it can be useful to change instance names when updating installed network configurations, this capability is available for any component that can be instantiated multiple times (that is, a Solution Model), whether or not it is part of a network configuration.

11.1.4.2 Example Code for the Instance Name Functional Companion

For the full source code of the Instance Name Functional Companion, see Example B–5 on page B-22.

11.1.4.3 Setup for the Instance Name Functional Companion

This section describes what you must do before using the Instance Name Functional Companion.

Steps

1. After optionally modifying it, compile and install the Instance Name Functional Companion.

   The full source code of the Instance Name Functional Companion appears in Example B–5 on page B-22.

2. To display the instance names in the Summary window of the runtime Oracle Configurator, verify that DISPLAY_INSTANCE_NAME is set correctly in the CZ_DB_SETTINGS table.

   For more information, see: *Oracle Configurator Implementation Guide*, for a description and Section 2.1.2, "Configure New Items" on page 2-5.

3. In Oracle Configurator Developer, choose the component that is to be configured (probably a BOM Model), and ensure that it can be instantiated multiple times.

   For an imported BOM Model, the **Maximum Instances** must be unspecified, or greater than the **Minimum Instances**. For more information, see the *Oracle Configurator Developer User's Guide*.

   For a Component or Developer Model the **Instances** attribute must be set so that the **Maximum** is unspecified, or greater than the **Minimum Quantity**.

4. In Oracle Configurator Developer, modify your component to add a Text Feature named `InstanceName`.

   Do not define any **Text Value** for the Feature.

5. Define a Functional Companion rule that associates the Instance Name Functional Companion with your component, and set the **Definition** of the Functional Companion rule as Table 11–9 shows:

Table 11–9 Rule Definition Attribute and Definition Value

Rule Definition Attribute	Definition Value
Base Component	The root node of the Model.
Type	Event-Driven
Implementation	Java
Program String	The name of the Java class that implements the Functional Companion. For the example, this is `ChangeInstanceName` (Example B–5 on page B-22).

6. Perform the usual procedure for testing a Functional Companion, including generating the Active Model and using the Test/Debug module.

11.1.4.4 Effects of the Instance Name Functional Companion

The Instance Name Functional Companion extends the CIO class `FunctionalCompanion`, and implements its method `initialize()`. This means that the code executes when the current configuration session starts.

First, a String variable named `textFeature` is set to the value `InstanceName`.

The `initialize()` method calls:

```
mUi = this.getRuntimeNode().getConfiguration().getUserInterface();
```

This creates a `IUserInterface` object for the configuration associated with the currently selected runtime component.

Then the `initialize()` method registers a listener for the `POST_VALUE_CHANGE` event:

```
mUi.addUserInterfaceEventListener(IUserInterfaceEvent.POST_VALUE_CHANGE, this);
```

This causes handling of the event to occur after the end user has changed the value of the node--in this case, the Text Feature named `InstanceName`.

Then the Functional Companion implements the `handleUserInterfaceEvent()` method of the listener interface `IUserInterfaceEventListener`, which the Functional Companion implements.

This method:

- Uses `IUserInterfaceEvent.getUiNode()` to get the node whose value changed (the event being listened for)
- Checks that it is a Text Feature (`TEXT_FIELD`) and that its name matches the value of `textFeature` (in this example, the value is `InstanceName`).

Then the String `inputText` is set to the value entered for the Text Feature node.

Next, a `Component` object `c` is set to the current runtime component on the current UI window of the `IUserInterfaceEvent` object associated with the current configuration. `Component` is an interface that can refer to either a Component or a BOM Model. In this example, the runtime component is one of the instances of the Component or BOM Model that the Functional Companion was associated with.

Finally, `Component.setInstanceName()` is called to set the name of the instance to the value of `inputText`, which is the value entered by the end user.

In the runtime Oracle Configurator, multiple component instances are named in the navigation tree by combining the name of the Component or BOM Model with the Instance Header ID (CZ_CONFIG_ITEMS.INSTANCE_HEADER_ID). The Text Feature field InstanceName is empty. When the end user types text into the field and enters the new text value (by clicking outside the field or pressing the Enter key), the event listener for the Functional Companion:

- Detects the change.
- Gets the new value.
- Applies the value to the navigation tree as the new name of the instance node.

11.1.4.5 Modify the Instance Name Functional Companion

For information on using the classes and methods of the Configuration Interface Object (CIO), see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

If you modify your Model in accordance with the steps in Setup for the Instance Name Functional Companion on page 11-18, then you do not need to modify the Instance Name Functional Companion. If you use a different name for the Text Feature named `InstanceName`, then you need to modify the value of the variable `textFeature`, as indicated in the following code fragment:

```
String textFeature = "InstanceName";
```

It is also possible to change the name of an instance programatically, without obtaining the text of the new instance name from the end user. You can use the `setInstanceName()` method with a string value that you set in your code, rather by getting it from a Text feature.

11.1.5 Using Configuration Attributes with Install Base (IB)

You can adapt the **IBAttribute Functional Companion** to allow end users to collect attributes of configured components.

The IBAttribute Functional Companion is recommended, but not required for implementing the updating of installed configurations of Container Models.

This section describes a Functional Companion and associated methodology for using Configuration Attributes.

11.1.5.1 Purpose of the IBAttribute Functional Companion

The IBAttribute Functional Companion enables an end user of the runtime Oracle Configurator to collect the values of attribute Features that have been set on the

trackable children of a runtime component instance. When the configuration is saved, the configuration attribute values are automatically written to the pending transactions for Oracle Install Base.

11.1.5.2 Example Code for the IBAttribute Functional Companion

For the full source code of the IBAttribute Functional Companion, see Example B–3 on page B-12.

11.1.5.3 Setup for the IBAttribute Functional Companion

1. After optionally modifying it, compile and install the IBAttribute Functional Companion. The full source code of the IBAttribute Functional Companion appears in Example B–3 on page B-12.
2. For trackable items, map extended attributes in Install Base to configuration attribute Features and Properties in Configurator Developer.

   For more information, see Section 10.6.4, "Map Install Base Extended Attributes to CZ Attributes in Developer" on page 10-22.
3. Define a Functional Companion rule that associates the IBAttribute Functional Companion with your Model, setting the **Definition** of the Functional Companion rule as Table 11–10 shows:

Table 11–10 Rule Definition Attribute and Definition Value

Rule Definition Attribute	Definition Value
Base Component	The component whose instances need their attributes collected. The Functional Companion searches all descendants of the component for attributes.
Type	Event-Driven
Implementation	Java
Program String	The name of the Java class that implements the Functional Companion. For the example, this is `CZIBAttributeFuncComp` (Example B–3 on page B-12).

4. Perform the usual procedure for testing a Functional Companion. This includes generating the Active Model and using the Test/Debug module.

   The IBAttribute Functional Companion uses AppsContext for connecting to the database. This requires a login that generates an ICX session ticket. Therefore, your initialization message must pass the parameters `database_id` (instead of

alt_database_name) and icx_session_ticket (instead of gwyuid, user, and pwd). In order to use the Test button in Configurator Developer, you must log in with an Oracle Applications ID and password.

11.1.5.4 Effects of the IBAttribute Functional Companion

The IBAttribute Functional Companion extends the CIO class AutoFunctionalCompanion, and implements its method onLoad(). This means that the code is executed whenever a trackable instance is loaded. For each trackable BOM child of the instance, the Functional Companion:

- Collects configuration attribute information from the Properties.
- Finds the attribute Feature.
- Associates the attribute with the RuntimeNode.
- Propagates the attribute to the node's children based on the propagation mode. The IBAttribute Functional Companion associates attributes with a RuntimeNode by using the public class Attribute, which is part of the CIO"

Finally, the attribute values are saved to the CZ_CONFIG_EXT_ATTRIBUTES table when the configuration is saved; this table is a temporary holder for the attribute values while the attribute de-tails are pending for acceptance in Install Base. At this point, the attributes become part of the transaction details in the transaction that is pending for acceptance into Oracle Install Base. When the transaction is accepted (by installation through SFM), the configuration attribute values become part of the installed instance.

11.1.5.5 Modify the IBAttribute Functional Companion

For information on using the classes and methods of the Configuration Interface Object (CIO), see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

If you modify your Model in accordance with the steps in Section 11.1.5.3, "Setup for the IBAttribute Functional Companion" on page 11-22, then you do not need to modify the IBAttribute Functional Companion.

When modifying your Model, you must follow the conventions for naming Properties described in *Oracle Configurator Methodologie*s chapter on configuration attributes for the IBAttribute Functional Companion to be able to collect values for the attribute Features. To use different Property names, you must modify the character strings used to match Property names, so that they match your own Property names. The following example--Example 11–1--shows typical matching strings.

Example 11–1 Strings for Property Names in the Functional Companion

```
...
  private void getAttributes(RuntimeNode node) {
...
        if (prop.getName().startsWith("ATTR_")) {

...
        int beginningIndex = new String("ATTR_"). length();
    StringTokenizer tokens = new StringTokenizer(name.substring(beginningIndex),
"_", false);
...
        if (name.endsWith("PATH")) {
...
    } else if (name.endsWith("MODE")) {

...
    } else if (name.endsWith("NAME")) {
...
    } else if (name.endsWith("GROUP")) {
...
      if (prop.getName().startsWith("ATTR_NAME")) {
...
      } else if (prop.getName().startsWith("ATTR_GROUP")) {
```

11.2 Using the Oracle Configuration Interface Object

For information about using this API, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

Main topics in this section include:

- Section 11.2.1, "Update Installed Configurations" on page 11-25
- Section 11.2.2, "Handling Interface Events" on page 11-28
- Section 11.2.3, "Automatic Behavior for Configurations" on page 11-31

11.2.1 Update Installed Configurations

In addition to single-instance configurations, Oracle Configurator can create and maintain configurations that consist of multiple instances of trackable configurable components. That is, the instances can be joined into a network by creating connections. For more information about instance configurations, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

Each component instance has a unique Instance Header ID identifier, stored as CZ_CONFIG_ITEMS.INSTANCE_HDR_ID. You can save, restore, revise, and save again a component instance. Each saved instance revision has a non-unique Instance Revision Number identifier, stored as CZ_CONFIG_ITEMS.INSTANCE_REV_NBR. The unique combination of Header ID and Revision Number identifies an instance record. For more information on identifying configurations, see the *Oracle Configurator Implementation Guide*.

Configurations can include multiple component instances, and you can configure the same component instance in multiple configuration sessions.

In Oracle Install Base, you can track the baseline and revised component instances in a configuration. For information about how to determine differences between the baseline for a component in Install Base and the revised instance of that component, see Section 11.2.1.1, "Identify Changes to a Configuration" on page 11-25.

For information on creating configuration models that support configuration updates, see Section 10.2, "Create the Configuration Model" on page 10-16.

11.2.1.1 Identify Changes to a Configuration

You can use the set of methods appearing in Table 11–11, "Methods for Identifying Changes Against the Baseline" on page 11-26 to identify the types of changes that occurred during a configuration session between the baseline revision of the trackable component instance (as installed in Oracle Install Base when the

configuration session began) and the current component instance (which is being updated).

The methods appearing in Table 11–11, "Methods for Identifying Changes Against the Baseline" are all members of the `IRuntimeNode` interface.

Table 11–11 Methods for Identifying Changes Against the Baseline

Method	Returns TRUE If...
`isChanged()`	Any of the other methods listed here return TRUE.
`isAddChanged()`	This node has been added in this session, meaning that it is not present in the installed revision.
`isDeleteChanged()`	This node has been deleted in this session and was present in the installed revision.
`isValueChanged()`	This node's value has changed from that of the installed revision.
`isTargetChanged()`	This Connector's target has changed from that of the installed revision.
`isAttributeChanged()`	Any of this node's configuration attributes have changed from those of the latest installed revision.
`isLocationChanged()`	This node's Location or Location Type Code has changed from that of the installed revision.
`isNameChanged()`	This instance's name has changed from that of the installed revision.
`isChildChanged()`	Any descendant of this node has changed from that of the installed revision. You control which types of nodes are examined for the `isChildChanged()` method by setting a value for the `deltaMode` switch of the Oracle Configurator Servlet property `cz.activemodel`. Setting the switch to `ALL` causes `isChildChanged()` to examine all descendants of the current node. Setting the switch to `BOM` restricts the examination to BOM nodes only. An examination of BOM nodes may be somewhat faster. For information about setting this property, see the *Oracle Configurator Installation Guide*.

Examples

For a fragmentary example of how to use one of these methods, see Example 11–2, "Identifying a Change Against the Baseline".

Example 11–2 Identifying a Change Against the Baseline

```
...
public boolean wasAdded(RuntimeNode node) {
    boolean added = node.isAddChanged();
    return added;
}
...
```

For a full example of how to use all of these methods, see Example B–2, "Setting the Line Type (LineTypeFunc.java)" on page B-5. For additional explanation, see also Section 11.1.3, "Specify the Line Type" on page 11-9.

11.2.1.2 Accessing Discontinued Items

To access both current and discontinued items in a Telecommunications Services Ordering solution, you must use the following alternate signatures for these methods:

- `RuntimeNode.getChildByName(String name, int type)`
- `RuntimeNode.getChildByID(int ID, int type)`
- `RuntimeNode.getChildByPersistentID(int ID, int type)`

Each signature includes a second argument, type. The possible values for type are:

- `Runtime Node CURRENT_OR_DISCONTINUED_CHILD`
- `Runtime Node CURRENT_CHILD`
- `Runtime Node DISCONTINUED_CHILD`

The default type is CURRENT_CHILD.

11.2.1.3 Accessing Instances

To access trackable component instances in a Telecommunications Services Ordering solution, use one of the following methods:

- Use `Component.getInstanceNumberLong()` to get the Instance Header ID of a trackable component instance. If the component instance is not trackable, this method returns the Instance Number. If the component is a mandatory component (Minimum Instances and Maximum Instances are both 1), this method returns 1.
- Use `ComponentSet.getChildByInstanceNumber(int instNum)` to get a trackable component instance by passing an Instance Header ID.

- If you have set the name for a component instance--by using `Component.setInstanceName(String newName)`--then you can get that instance by the name you set, by using `ComponentSet.getChildByName(String userSetName)` on the `ComponentSet` that includes the instance.
- In order to access both current and discontinued component instances, use the following alternate signature: `ComponentSet.getChildByInstanceNumber(int instNum, int type)`. For type, specify one of the values listed under Section 11.2.1.2, "Accessing Discontinued Items" on page 11-27 (such as CURRENT_CHILD).

11.2.1.4 Determining Editability

When performing updates in a network of trackable component instances in a TSO solution, you may need to activate a neighboring component, and determine whether the component is editable. You can use the method `IRuntimeNode.isEditable()` to determine whether a component is editable, as shown in Section B.5, "Determining Editability" on page B-24. This example also shows how to use the event `IUserInterfaceEvent.POST_ACTIVATE_ACTION` to determine when an activation event takes place.

11.2.2 Handling Interface Events

Several TSO-specific events occur when Oracle Configurator updates an installed configuration of network components. This section describes how to hand these events.

For general information on interface events not specific to a TSO solution, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

The ability to update a network of configured components introduces the following events in the `IUserInterfaceEvent` interface:

- ON_NAVIGATE
- POST_ACTIVATE_ACTION

Handling these events requires you to register a listener. For more information on registering listeners, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

For background on configuration updates, see Chapter 2, "TSO Business Process".

ON_NAVIGATE

Handling the ON_NAVIGATE event lets you alter the window that the end user is about to see.

You should consider the situations under which users can navigate to a window. For example, if the window that the end user navigates to might be a read-only component, then you may want to render the UI controls in a manner that is different from that for an editable component.

The UI Server notifies ON_NAVIGATE event listeners when a window navigation is about to occur. The details about that notification follow.

- When the configuration session starts, the UI Server notifies ON_NAVIGATE event listeners after processing of both the POST_RESTORE_CONFIGURATION and POST_NEW_CONFIGURATION events has occurred. After the listeners for ON_NAVIGATE have been invoked, any changes that your Functional Companion makes appear when the new window appears.
- During the configuration session, the UI Server notifies ON_NAVIGATE event listeners after processing for all other event listeners has occurred and if a navigational window change needs to occur. Window changes can happen when either:
  - An end user clicks a navigation button.
  - A Functional Companion requests navigation.
- When the configuration session terminates, the UI Server notifies ON_NAVIGATE event listeners under one of the following scenarios:

  First scenario:

  1. After the PRE_SAVE_CONFIGURATION events have been processed.
  2. If your Functional Companion listening for PRE_SAVE_CONFIGURATION events aborts the **save** operation by generating a FuncCompMessageException.
  3. If a navigational window change needs to occur.

  Second scenario:

  1. After the PRE_CANCEL_CONFIGURATION events have been processed.
  2. If your Functional Companion listening for PRE_SAVE_CONFIGURATION events aborts the **cancel** operation by generating a FuncCompMessageException.

3. If a navigational window change needs to occur.

After either of these scenarios, the changes appear in the UI result set, so that window rendering can occur.

Note: The UI Server does not notify ON_NAVIGATE event listeners after a POST_SAVE_CONFIGURATION event.

You must distinguish the purpose of ON_NAVIGATE from that of POST_NAVIGATION_EVENT_EXECUTION. The ON_NAVIGATE event lets you customize the window that is about to appear, while POST_NAVIGATION_EVENT_EXECUTION lets you validate the window that the end user is leaving or choose which window to navigate to next. The handling of the POST_NAVIGATION_EVENT_EXECUTION event can be used to:

- Cancel navigation if:
  - Certain UI controls do not have any values, such as a text field that should contain the name of the component but does not.
  - Certain UI controls do not have the desired values.
  - The format of the values is not correct.
- Decide which window to navigate to based on the current state of the model.

POST_ACTIVATE_ACTION

Handling the POST_ACTIVATE_ACTION event lets you change the display state of controls on the current window.

The UI Server notifies POST_ACTIVATE_ACTION event listeners when either of these events happens:

- The end user clicks an **Activate Instance** UI control.
- A read-only component is activated through a Functional Companion *and* that component currently appears in the window.

Some details about that notification of POST_ACTIVATE_ACTION event listeners include:

- During the configuration session, the UI Server notifies POST_ACTIVATE_ACTION listeners under either of these conditions:
  - After all other event listeners are processed, but before the ON_NAVIGATE listeners.

  - If the UI Server is currently processing the event that activates a read-only component.
- If the configuration session is terminating, the UI Server does not notify POST_ACTIVATE_ACTION listeners.

11.2.3 Automatic Behavior for Configurations

The `onLoad( )` method of the AutoFunctionalCompanion is used when updating installed configurations of network components. For other methods of the AutoFunctionalCompanion, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

Table 11–12 Methods of AutoFunctionalCompanion

Method to Override	Executed...	Comments and Examples
`onLoad()`	When a runtime component loads.	A component loads when either an instance is added or a read-only runtime component is brought into the configuration.

12

Set Up Quoting

The main topics in this chapter are:

- Section 12.1, "Summary of Reconfiguration" on page 12-2
- Section 12.2, "Define Profile Options" on page 12-3
- Section 12.3, "Set Up the Lines Page" on page 12-6
- Section 12.4, "Enable or Disable Line-Level Actions or Table-Level Buttons" on page 12-13
- Section 12.6, "Design Tips" on page 12-16

Reference Material

This chapter describes setup tasks that you must perform to Oracle Quoting that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on functionality, setting up, or using Oracle Quoting, refer to:

- Section 3.2, "Quoting" on page 3-3
- *Oracle Quoting Implementation Guide*
- *Oracle Quoting User Guide*

Before You Begin

Before you can set up Oracle Quoting for use with the Oracle Telecommunications Service Ordering solution, you must set up Oracle Advanced Pricing and Oracle Configurator. For more information, see:

- Chapter 9, "Set Up Advanced Pricing"
- Chapter 10, "Set Up Configurator and Customize the Solution"
- Chapter 11, "Set Up Configurator Functional Companions"

12.1 Summary of Reconfiguration

To make use of reconfiguring services, you specify:

- Profile Options
- Lookup Codes
- Additional Line Types

Profile Options

To make use of reconfiguring services, specify the following profile options.

- ASO: Configurator URL
- ASO: Default Order Type
- ASO: Default Quote Status
- ASO: Default Order Status
- ASO: Quote Duration
- ASO: Default Order Quantity
- ASO: Use Network Container
- ASO: Allow Quantity Updates For Top Level Model Item
- ASO: Allow Quantity Updates For Component Items
- Sequential Numbering

  For more information, see Section 12.2, "Define Profile Options".

Lookup Codes

To make use of reconfiguring services, enable the following lookup codes.

- ADDIB for the lookup type ASO_LINE_ITEM_ACTION.

  Enabling this lookup code causes the **Add to Model Item From Install Base** option to appear in the Lines page Action Menu.

  For more information, see Section 12.4, "Enable or Disable Line-Level Actions or Table-Level Buttons".

- RECONFIG for the lookup type ASO_IB_ACTION_TYPES.

  Enabling this lookup code causes the **Reconfigure** option to appear as an option on the Search and Select: Install Base page.

For more information, see Section 12.4, "Enable or Disable Line-Level Actions or Table-Level Buttons".

- REMOVEPROD for the lookup type ASO_LINE_ACTION.

  Enabling this lookup causes the **Remove Product** option to appear in the Lines page.

  For more information, see Section 12.4, "Enable or Disable Line-Level Actions or Table-Level Buttons".

Additional Line Types

You can set up additional customized Line Types for reconfigurations.

For instructions on how to set up additional Line Types, refer to the steps for defining Transaction Types in the *Oracle Order Management Suite Implementation Manual*. For information on setting up the Line Type that ensures that orders pass into Service Fulfillment Manager, see the *Oracle Service Fulfillment Manager Implementation Guide*.

For more information on related TSO setup for Line Types, see:

- Section 5.1, "Ensure Application of Patch 2529922" on page 5-3
- Section 5.3, "Set Up Transaction Types" on page 5-4
- Section 11.1.3, "Specify the Line Type" on page 11-9

12.2 Define Profile Options

The following Oracle Order Capture and Order Management profile options pertain to the Oracle Telecommunications Service Ordering (TSO) functionality. For a complete list of mandatory Quoting profile options, see the *Oracle Quoting Implementation Guide*.

> **Note:** Profile options with an asterisk in Table 12–1, "Oracle Telecommunications Service Ordering Profile Options for Configurator" are mandatory for the Oracle Telecommunications Service Ordering functionality.

- **ASO: Allow Quantity Updates for Component Item**

  Determines whether users can change the quantity of a component.

You must set this option to No to prevent the user from changing the quantity of Component items of a Container model.

- **ASO: Allow Quantity Updates for Top Level Model Item**

  Determines whether users can change the quantity of a top-level model product. You must set this option to **No** to prevent users from changing the quantity of the network container.

  If set to Yes, users can change the quantity of top-level model products.

  If set to No, users cannot update the quantity of top-level model products.

  If null, defaults to Yes.

- **ASO: Configurator URL**

  Specifies the URL for the Configurator window.

  Mandatory if Oracle Configurator is used for product configurations.

  If null, there is no default value.

- **ASO: Default Install Base Relationship**

  Determines the default Install Base relationship for Relationship Type menu on the Install Base Relationships page.

  If set to **Component_Of**, the Relationship Type defaults to Component-of.

  If set to **Connected_To**, the Relationship Type defaults to Connected-To.

  If null, the default value is **Component_Of**.

- **ASO: Default Order Quantity**

  Determines the default quantity that appears in the QTY field of the Oracle Telecommunications Service Ordering window.

  If null, a default quantity of **1** applies.

  Must set to **1** if you are using a Container Model.

- **ASO: Default Order Type**

  Determines how the order will be processed in Oracle Order Management and Service Fulfillment Manager. You set up order types in Oracle Order Management. This must include workflow to pass order to Service Fulfillment Manager. For setup information of Order Type in Order Management, see Service Fulfillment Manager setup or Order Management setup.

  If null, no default value applies.

- **ASO: Default Quote Status**

  Determines the default status of a new quote.

- **ASO: Use Network Container**

  Determines if network container functionality is enabled.

  If the setting is Yes, you can reconfigure telecommunications services.

  If the setting is No, you cannot reconfigure telecommunications services.

  If null, the default is No.

- **Sequential Numbering**

  Determines if order numbers are automatically generated by Oracle Order Management or if users must enter them. For more information, see the *Oracle Order Management Suite Implementation Manual*.

  Note: Oracle Quoting only supports use of automatically generated order numbers.

  Table 12–1 shows you the Profile Options, Default Value, Application Level, and Suggested Setting if you are using TSO functionality.

Table 12–1 Oracle Telecommunications Service Ordering Profile Options for Configurator

Profile Option	Default Value	Application Level	Suggested setting if using Telecommunication Services Functionality
ASO: Allow Quantity Updates for Component Item	Yes	Site, Application	No
ASO: Allow Quantity Updates for Top Level Model Item	Yes	Site, Application	No
*ASO: Configurator URL	N/A	Site, Application, Responsibility, User	N/A
ASO: Default Install Base Relationship	Component-Of	Site, Application	Component-Of
*ASO: Default Ordered Quantity in OC UI	1	Site, Application, Responsibility, User	1
ASO: Default Order Status	Entered	Site	Booked

Table 12–1 Oracle Telecommunications Service Ordering Profile Options for Configurator (Cont.)

Profile Option	Default Value	Application Level	Suggested setting if using Telecommunication Services Functionality
*ASO: Default Order Type	N/A	Site, Application, Responsibility, User	N/A
*ASO: Default Quote Status	Draft	Site, Application	Draft
*ASO: Use Network Container	No	Site	Yes
Sequential Numbering		Site, Application, Responsibility	Always Used

12.3 Set Up the Lines Page

Oracle Quoting allows customization of certain aspects of the Lines page to meet your needs. The following list shows the suggested setup of the Lines page when using Oracle Telecommunications Service Ordering.

- Select
- Line
- Action (Line Type)
- Instance Name
- Product
- Description
- Additional Information
- UOM
- Quantity
- Recurring List
- Recurring Adjustment
- Recurring Net
- One Time Charges (Charges)

Steps

1. Expose the following columns.
   - Line Type
   - Instance Name
   - Charges

   For more information, see Section 12.3.1, "Display or Hide Columns".
2. Hide the following column.
   - Total Price

   For more information, see Section 12.3.1, "Display or Hide Columns".
3. Rename columns.

   For more information, see Section 12.3.3, "Edit Column Labels".

   Table 12–2 shows the original and suggested names column names on the Line Page.

Table 12–2 Suggested Renaming of Columns on the Line Page

Original Column Name	Suggested Name
Line Type	Action
Instance Name	Instance Name or Description
Unit List Price	Recurring List
Unit Adjustment Percent	Recurring Adjustment
Unit Selling Price	Recurring Net
Charges	One Time Charges

4. Change the column sequence.

   For more information, see Section 12.3.4, "Edit Column Sequence".

> **Note:** After you change the seeded values, you cannot automatically reset them back to their original definitions.

You can disable Line Actions that users cannot perform on reconfigured telecommunications service products. For more information, see Section 12.4,

"Enable or Disable Line-Level Actions or Table-Level Buttons". You can disable the following Actions:

- Split Line
- Add Service

Table 12–3 shows a mapping of the column name to the attribute name.

Table 12–3 Column/Attribute Mapping

Column Name	Attribute Name
Select	QOT_SELECT
Product	QOT_PRODUCT
Description	QOT_DESCRIPTION
Quantity	QOT_QUANTITY
UOM	QOT_UOM
Unit List Price	QOT_UNIT_LIST_PRICE
Unit Adjustment Percent	QOT_UNIT_ADJUST_PERCENT
Line Discount	QOT_LINE_DISCOUNT
Unit Selling Price	QOT_UNIT_SELLING_PRICE
Total Price	QOT_TOTAL_PRICE
Charges	QOT_CHARGES
Line Category	QOT_LINE_CATEGORY
Line Type	QOT_LINE_TYPE
Line (Line Number)	QOT_LINE
Instance Name	QOT_INSTANCE_NAME
Custom Columns	QOT_CUSTOM_COL1, QOT_CUSTOM_ COL2, QOT_CUSTOM_COL3
Tax Details	QOT_TAX_DETAILS

12.3.1 Display or Hide Columns

You can choose which seeded columns to display or hide on the Lines page and in the Lines section of the Overview page.

The following columns are mandatory:

- Select
- Product
- Description
- UOM
- Quantity

> **Note:** Although you can deselect the Node Display check box for the mandatory columns, they will still appear in the Quoting user interface. You cannot disable these columns.

The following seeded columns are optional:

- Unit List Price
- Unit Adjustment Percent
- Line Discount

> **Note:** If you set profile option **ASO: Calculate Price** to Manual, the Line Discount field does not appear even if you have enabled it and the user has the ability to perform price overrides.

- Unit Selling Price
- Total Price
- Charges
- Line Category
- Line Type
- Line Number
- Instance Name
- Link to Customer Page
- Tax Details

12.3.2 Hide a Seeded Column

You can hide a seeded column if you do not want a column to appear on the Lines page or in the Lines section of the Overview page.

Module

Oracle Applications (Forms)

Responsibility

Applications Developer Common Modules

Steps

1. Navigate to **Define Regions**.

   The Define Regions form appears.

2. From the View menu, choose **Query by Example > Enter**.

3. To hide or unhide a column on the Lines page, enter `QOTLINEDETAILS` in the Region ID field.

4. In the Region Name column, enter `QUOTELINEDETAILS`.

5. From the View menu, choose **Query by Example > Run**.

6. Select **Region Items**.

   The Region Items form appears.

7. To hide a seeded column, deselect the Node Display check box for the corresponding column.

   Selecting this check box displays the column.

   Note: Although you can deselect the Node Display check box for the mandatory columns, they will still display in the Quoting user interface. You cannot disable these columns.

8. Click the Save icon.

   A confirmation message appears with your request identification.

9. Bounce Apache.

   When you login the next time, the changes take effect.

12.3.3 Edit Column Labels

You can edit default text labels for each column.

Table 12–4 shows the recommended columns and labels for the Telecommunications Services Ordering solution.

Table 12–4 Recommended Columns and Labels

Seeded Column Label	Change Column Label to...
Line Type	Provisioning Action.
Instance Name	No change necessary. Use this field to enter the name of the configured instance.
Unit List Price	Recurring Price List. The price list includes the recurring price of a service. The recurring price of the service displays here.
Unit Adjustment Percent	Recurring Adjustment.
Charges	One Time Charges. Displays the applicable fixed charges for the quote line.

Module

Oracle Applications (Forms)

Responsibility

Applications Developer Common Modules

Steps

1. Navigate to **Define Regions**.

   The Define Regions form appears.

2. From the View menu, choose **Query by Example > Enter**.
3. In the Region ID column, enter `QUOTELINEDETAILS`.
4. In the Region Name column, enter `QUOTELINEDETAILS`.
5. From the View menu, choose **Query by Example > Run**.
6. Select **Region Items**.

   The Region Items form appears.

7. To change a column's label, enter a new value in the Long Label field of the corresponding line.

8. Click the Save icon.

   A confirmation message appears with your request identification.

9. Bounce Apache.

   When you login the next time, the changes take effect.

12.3.4 Edit Column Sequence

You can specify the order in which columns appear.

Module

Oracle Applications (Forms)

Responsibility

Applications Developer Common Modules

Steps

1. Navigate to **Define Regions**.

   The Define Regions form appears.

2. From the View menu, choose **Query by Example > Enter**.

3. In the Region ID column, enter `QUOTELINEDETAILS`.

4. In the Region Name column, enter `QUOTELINEDETAILS`.

5. From the View menu, choose **Query by Example > Run**.

6. Select **Region Items**.

   The Region Items form appears.

7. To rearrange columns, change the values in the Sequence fields.

   The lowest sequence number is the first column on the Lines page.

8. Click the Save icon.

   A confirmation message appears with your request identification.

9. Bounce Apache.

   When you login the next time, the changes take effect.

12.4 Enable or Disable Line-Level Actions or Table-Level Buttons

You can enable or disable line-level actions and table-level buttons. This allows you to customize the actions that a user can perform on the Lines page.

You enable or disable the Actions menus on the Lines and Overview pages. You can also customize the buttons on the Lines page. Through customization, you determine which options appear.

> **Note:** To add new lookup codes to the lookup types, you must implement the logic. Oracle Quoting only supports the seeded out-of-box values.

Lookup codes control the menus. Table 12–5 shows the lookup type for action menus and buttons.

Table 12–5 Lookup Types for Action Menus and Buttons

Lookup Type	Description
ASO_LINE_ITEM_ ACTION	Determines which actions appear on the Lines page Actions menu.
ASO_LINE_ACTION	Determines which line-level actions appear on the Lines page.
ASO_QUOTE_ACTION	Determines which actions appear in the Overview page Actions menu.
ASO_IB_ACTION_TYPES	Determines which actions are available on the Search and Select: Install Base Product page.
ASO_PRODUCT_ SOURCE	Determines sources are available when performing a product search.
ASO_IB_ RELATIONSHIP_TYPES	Determines which Install Base relationship types are available.

Module

Oracle Applications (Forms)

Responsibility

Quoting Sales Manager

Prerequisites

The lookup code must be extensible.

Steps

1. Navigate to **Quick Codes**.
2. Select **View > Query by Example > Enter**.
3. To enable the **Add to Model Item From Install Base** action, enter **`ASO_LINE_ITEM_ACTION`** in the Type field.

   To enable the Reconfigure action, enter **`ASO_IB_ACTION_TYPES`** in the Type field.

   To enable the Remove Product action, enter **`ASO_LINE_ITEM_ACTION`** in the Type field.

   To enable the Connected-To option on the Relationships page, enter **`ASO_IB_RELATIONSHIP_TYPES`** in the Type field.

   To disable the Add Service or Split Line actions, enter **`ASO_LINE_ACTION`** in the Type field.
4. Choose **View > Query by Example > Run**.

   The record appears.
5. Locate the code to enable and select the Enabled check box:
   - **ADDIB**: Enable the Add to Model Item from Install Base action.
   - **RECONFIG**: Enable the Reconfigure action.
   - **REMOVEPROD**: Enable the Remove Product action.
   - **CONNECTED-TO**: Enable the Connected-To option.
6. To disable the lookup code, deselect the check box:
   - **SPLIT**: Disable Split Line action
   - **ADDSVC**: Disable Add Service action
7. Save your work.

12.5 Change Default Number of Rows

A user can search Install Base to select product instances for reconfiguration. Based on the specified search criteria in the **Search and Select: Installed Base Product** page, a list of search results appear.

If the number of rows resulting from the search exceeds one page, the user is restricted to selecting items only from one search results page. The user can select products from other pages for reconfiguration by using the **Add To Model from Installed Base** line action. For more information, see Section 14.6, "Add to Container Model from Install Base" on page 14-7.

You can also increase the number of rows that appear after a search so that the user can select more products for reconfiguration from one search results page. This section describes how to change the default number of rows appearing in the Oracle Quoting application.

Module

Oracle Applications (Forms)

Responsibility

System Administrator

Navigation

Profile > System

Steps

1. On the **Find System Profile Values** form, select the **Application** check box
2. In the Application field, enter **Oracle Quoting**.
3. In the Profile field, enter **JTF_PROFILE_DEFAULT_NUM_ROWS**.
4. Click the Find button.

   The System Profile Values page appears.
5. In the Application column, enter the number of default rows that you want to display.
6. Save your work.
7. Bounce **apache** with no clear cache to view the change in Quoting.

12.6 Design Tips

Consider **either** of the following for the default Line Type:

- An Oracle Configurator Functional Companion must always set the default Line Type.
- The default Line Type should be blank in Quoting, and the Order Management defaulting rules in the order (as opposed to the quote) set the default Line Type.

Use the selectable columns feature to show the Line Type on the Quoting lines page and to rename columns, such as renaming Line Type to Action.

Use lookups to disable line and table-level actions that do not apply to the given implementation, such as Split Line, Add Service, ATP check, or Trade In option in Install Base searches.

13

Set Up Service Fulfillment Manager

The main topics in this chapter are:

- Section 13.1, "Create SFM Enabled Order Management Workflow Header Process" on page 13-3
- Section 13.2, "Create SFM Enabled Order Management Workflow Line Process"
- Section 13.3, "Add Function for Menu Notifications" on page 13-5
- Section 13.4, "Create a PL/SQL Procedure That Updates a Certain Workitem Parameter" on page 13-6
- Section 13.5, "Create Workflow Process" on page 13-7
- Section 13.6, "Define Work Item" on page 13-9
- Section 13.7, "Map Work Item to Item and Action (Line Transaction Type) Combination" on page 13-10
- Section 13.8, "Design Tips" on page 13-11

Reference Material

This chapter describes setup tasks that you must perform to Oracle Service Fulfillment Manager that are specific to the Oracle Telecommunications Service Ordering (TSO) solution. For more information on functionality, setting up, or using Oracle Service Fulfillment Manager, refer to:

- Section 3.6, "Service Fulfillment Manager" on page 3-12
- *Oracle Service Fulfillment Manager Implementation Guide*

Before You Begin

Before you can set up Oracle Service Fulfillment Manager for use with the Oracle Telecommunications Service Ordering solution, you must set up Oracle Install Base and Oracle Order Management. For more information, see Chapter 8, "Set Up Install Base".

13.1 Create SFM Enabled Order Management Workflow Header Process

Module

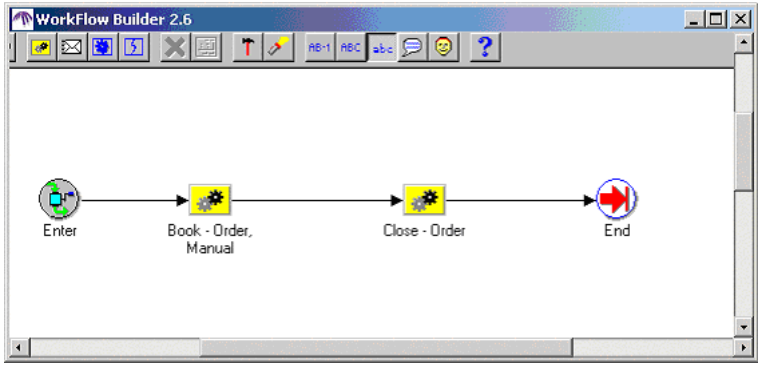
Workflow Builder

Steps

1. Open the **OM Order Header** workflow item type (OEOH).
2. Open the **Service Fulfillment Header** workflow item type (XDPOMOH).
3. In the item type OEOH, open the **Order Flow - Generic** process.

   Figure 13–1, "Example Order Flow - Generic Process" shows an example of the order flow process.

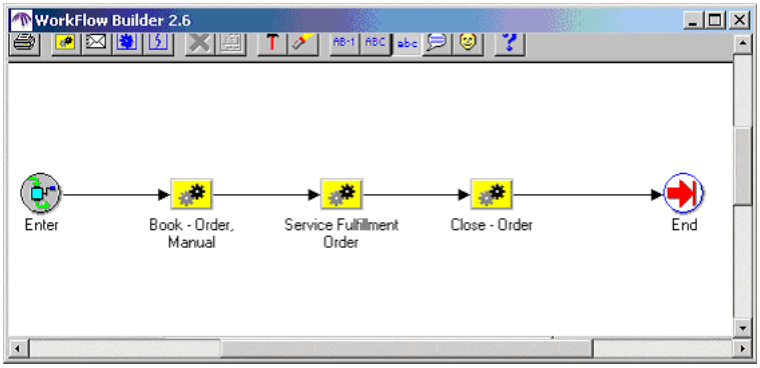
Figure 13–1 Example Order Flow - Generic Process



4. Save this process with a new name **Order Flow - Service Fulfillment**.
5. Drag and drop the **Service Fulfillment Order** process from the item type XDPOMOH into **Order Flow - Service Fulfillment** between the **Book - Order, Manual** and **Close - Order** processes.

Figure 13–2 shows the positioning of the **Service Fulfillment Order** process in the process.

Figure 13–2 Order Flow - Service Fulfillment



6. Save your work.

13.2 Create SFM Enabled Order Management Workflow Line Process

Module

Workflow Builder

Steps

1. To customize the Workflow, change the Access Level to 100:
   - Choose Help > About Oracle Workflow Builder.
   - In the Access Level field, enter 100.
   - Click OK.
2. Open the **OM Order Line** workflow item type (OEOL).
3. Open the **Service Fulfillment Line** workflow item type (XDPOMOL).
4. In item type **OEOL**, open the **Line Flow- Generic** process.
5. Save the **Line Flow- Generic** process with a new name **Line Flow - Service Fulfillment**.

6. Drag the activity **Service Fulfillment (For Activation Required Items)** from item type XDPOMOL and drop it between **Ship - Line, Manual** and **Fulfill - Deferred**.
7. Drag the activity **Installed Base** from item type XDPOMOL and drop it between **Fulfill** and **Invoice Interface - Line**.
8. Save your work.

13.3 Add Function for Menu Notifications

In this procedure you are adding the Workitem Notification Redirection to the menu for notifications. This procedure assumes that you have not already added Workitem Notification Redirection to the menu. This function lets you view error notifications in HTML. For more information, see Section 15.6, "Manage Failure Notifications in HTML" on page 15-7.

Module

Oracle Applications (Forms)

Responsibility

System Administrator

Navigation

Applications > Menu

Steps

1. Select Menu and click Open.

   The Menus window opens.
2. In the Menu field, query **FND_WFADMIN_NEW**.
3. To add a line, click the New icon.
4. On the new empty line, keep the Prompt and Submenu fields empty.
5. In the Function field, click the List of Values.

   The Function window opens.
6. Enter the Search criteria, **%redirect%**.
7. Select **Workitem Notification Redirection**, and click OK.

Both the Function and Description fields automatically populate.

8. Click the Save icon.
9. When the message about recompiling menus appears, click OK.

See Also

The end-user procedure that uses the preceding setup is Section 15.6, "Manage Failure Notifications in HTML" on page 15-7

13.4 Create a PL/SQL Procedure That Updates a Certain Workitem Parameter

The following information is a specific example that you can consider for a reference. It is not a requirement to you create your PL/SQL Procedure in this manner.

Module

Your favorite PL/SQL Editor

Step

Create the following package in your instance.

The procedure in this package updates the work item parameter ACTIVE_FLAG to **Y**. You defined ACTIVE_FLAG as an extended attribute for your Service Fulfilment Items in Section 13.7, "Map Work Item to Item and Action (Line Transaction Type) Combination" on page 13-10.

```
CREATE OR REPLACE PACKAGE XXMACD_SFM_UTILS AS

  PROCEDURE SET_ACTIVE(itemtype      IN VARCHAR2,
                       itemkey       IN VARCHAR2,
                       actid         IN NUMBER,
                       funcmode      IN VARCHAR2,
                       resultout    OUT VARCHAR2);
END;
/
CREATE OR REPLACE PACKAGE BODY XXMACD_SFM_UTILS AS
  PROCEDURE SET_ACTIVE(itemtype      IN VARCHAR2,
                       itemkey       IN VARCHAR2,
                       actid         IN NUMBER,
                       funcmode      IN VARCHAR2,
                       resultout    OUT VARCHAR2) IS
```

```
    l_wi_instance_id    NUMBER ;
  BEGIN
  -- RUN mode - normal process execution
  --
   IF (funcmode = 'RUN') THEN
      l_wi_instance_id := WF_ENGINE.GetItemAttrNumber
                               (itemtype => itemtype ,
                                itemkey  => itemkey ,
                                aname    => 'WORKITEM_INSTANCE_ID');


      XDP_ENGINE.SET_WORKITEM_PARAM_VALUE
                              (P_WI_INSTANCE_ID    => l_wi_instance_id,
                               P_PARAMETER_NAME    => 'ACTIVE_FLAG',
                               P_PARAMETER_VALUE   => 'Y');

      resultout  := 'COMPLETE';
      RETURN;
    ELSE
      resultout := xdp_om_util.HandleOtherWFFuncmode(funcmode);
      RETURN;
    END IF;
  EXCEPTION
    WHEN OTHERS THEN
     wf_core.context('XXMACD_SFM_UTILS','SET_ACTIVE', itemtype, itemkey, actid,
NULL);
      RAISE;
  END SET_ACTIVE;
END XXMACD_SFM_UTILS;
```

13.5 Create Workflow Process

The following example is an option. It is not a requirement.

This example assumeS sample names for:

- **Item Type**: SFMTEST - SFM Test Item Type
- **Function**: Update WI Active Flag
- **Procedure**: XXMACD_SFM_UTILS.SET_ACTIVE
- **Process**: SFM MACD Test WorkItem Process

Module

Workflow Builder

Steps

1. Create a new Item Type:

   For example: **SFMTEST - SFM Test Item Type**.

2. Associate the following three previously defined Service Fulfillment Manager item types to the new item type:

   – SFM Lookup Codes

   – SFM Standard

   – Standard

3. Create a Function.

   For example: **Update WI Active Flag**.

4. Map the function to procedure XXMACD_SFM_UTILS.SET_ACTIVE that you created in Section 13.4, "Create a PL/SQL Procedure That Updates a Certain Workitem Parameter" on page 13-6.

5. Map Error Item Type to XDPWFSTD.

6. Map Error Process to XDP_ERROR_PROCESS.

7. Define a process.

   For example: **SFM MACD Test WorkItem Process**.

8. Drag the function **Update WI Active Flag** into the process.

9. Drag the functions **Start** and **End** from Item Type Standard and into the process.

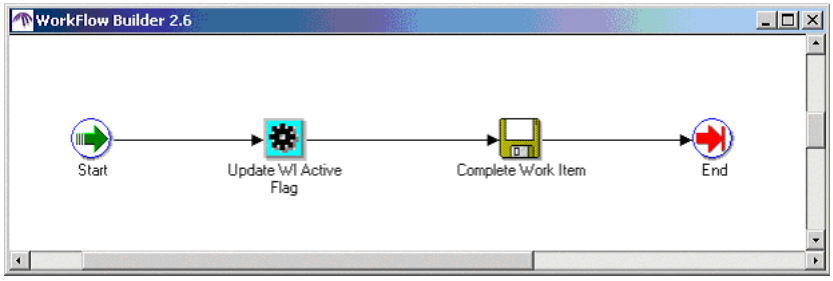
10. Drag the function **Complete Work Item** from Item Type SFM Standard into the process.

11. Define the flow between the functions as the following picture shows.

    Figure 13–3 shows the Start icon on the left with a Connector pointing to Update WI Update Flag. From the Update WI Update Flag, a Connector points

to Complete Work Item. From Complete Work Item, a Connector points to the End.

Figure 13–3 Flow Between the Functions



12. Save your work.

13.6 Define Work Item

The following example is an option. It is not a requirement.

This example assumeS sample names for:

Module

Oracle Applications (Forms)

Responsibility

SFM System Administrator

Prerequisites

Section 13.4, "Create a PL/SQL Procedure That Updates a Certain Workitem Parameter" on page 13-6

Section 13.5, "Create Workflow Process" on page 13-7

Navigation

Setup > Service Definition > Work Items

Steps

1. For the work item, specify Display Name (Set Active), Internal Name, Version and Begin Date.

2. From the list, choose **Work Item Type User-Defined Workflow**.
3. From the Item Type list of values, choose **SFM Test Item Type** that you created in Section 13.5, "Create Workflow Process" on page 13-7.
4. From the Process list of values, choose **SFM MACD Test WorkItem Process** that you created in Section 13.5, "Create Workflow Process" on page 13-7.
5. Specify an Item Key for this Work Item.

   For example, an Item Key could be **SFMMACD**. The Item Key value becomes the prefix for the Workflow Item Key.

   The value that you specify in this step will be the workflow item type defined in the section.
6. Navigate to the Parameters tab.
7. Choose the parameters **Active**, **Phone**, and **Primary**.

   These parameters are extended attributes that you defined in Section 8.2, "Create Extended Attributes" on page 8-3.

13.7 Map Work Item to Item and Action (Line Transaction Type) Combination

You can attach one or more work items or tasks to a combination of item and action.

Module

Service Fulfillment Manager

Responsibility

SFM Administrator

Navigation

Setup > Service Definition > Service Actions

Prerequisites

You must have created the work item, the item, and the action.

Steps

1. For each of the Service Fulfillment items that you defined according to standard setup for Service Fulfillment Manager, navigate to the Actions tab and map the Line Transaction Types **ADD** and **CHANGE** to the item.

   You created the preceding Line Transaction Types in Section 5.3.1, "Set Up Line Transaction Type" on page 5-4.

   For more information on setting up Service Fulfillment items, see *Oracle Service Fulfillment Manager Implementation Guide.*

2. Next, navigate to the Work Item for Action tab and map Work Item Set Activate to the two Item Action combinations.

13.8 Design Tips

Some tips relating to Service Fulfillment Manager include:

- An attribute that is mandatory in Service Fulfillment Manager also needs to be mandatory in Configurator Developer. For more information about configuration attributes, see *Oracle Configurator Methodologies*.
- If Service Fulfillment Manager fails to accept the order after booking the order in Quoting, then Quoting never receives the error message. In Order Management's Sales Order form, these errors have a status of **Provisioning Failed**. To see the status of these provisionable order lines, you must go to Order Management's Sales Order form.
- If Service Fulfillment Manager fails to accept the order after booking the order in Quoting, you must progress the order from Order Management's Sales Order form.
- The Bill of Materials routing includes the definition of the **activation sequence of lines**. When Service Fulfillment Manager receives a provisionable model with provisionable components, Service Fulfillment Manager provisions the model first and its components.
- The model developer must ensure that:
  - Every provisionable item includes a mapped Line Type in Service Fulfillment Manager.
  - The item and Line Type combination maps to a Workitem.

- When order booking occurs, Service Fulfillment Manager receives all the mandatory Workitem parameters (also known as **extended attributes**) with a value.

Part III

Using and Administering the Solution

This part contains the following chapters:

- Chapter 14, "Using the Solution"
- Chapter 15, "Administering the Solution"

14

Using the Solution

Topics in this chapter include:

- Section 14.1, "Introduction" on page 14-2
- Section 14.2, "Create a New Quote" on page 14-2
- Section 14.3, "Add Products to the Quote" on page 14-4
- Section 14.4, "Reconfigure an Existing Service" on page 14-5
- Section 14.5, "Add Products to Quote from Install Base" on page 14-6
- Section 14.6, "Add to Container Model from Install Base" on page 14-7
- Section 14.7, "Remove Unchanged Components" on page 14-8
- Section 14.8, "Remove Lines" on page 14-9
- Section 14.9, "Place an Order" on page 14-10

See Also

For administration and maintenance-oriented tasks, see Chapter 15, "Administering the Solution".

14.1 Introduction

You need to create a new quote when your customer wants to:

- Order a new service. For more information, see Section 14.2, "Create a New Quote" on page 14-2.
- Change or reconfigure their existing service. For more information, see Section 14.4, "Reconfigure an Existing Service" on page 14-5.

Reference

For detailed information about working with quotes, see *Oracle Quoting User Guide*.

14.2 Create a New Quote

When you create a new configuration for your customer, you create either a totally new quote or a quote from a template. In either case, you:

- Specify a name for the quote
- Specify the customer's name and location by performing a search.
- Change default values, if necessary.
- Select the appropriate Order Type that allows the process to flow into Service Fulfillment Manager.

If you are modifying an existing service, see Section 14.4, "Reconfigure an Existing Service" on page 14-5.

Module

Oracle Sales Online

Navigation

Quotes tab

Steps

1. On the Quotes page, click **Create Quote**.

Note: If the **ASO: Automatic Numbering** profile option is set to **No** at the appropriate level, you can enter a quote number which will be unique across all quotes. If this profile option is set to **Yes**, the system will automatically assign a number.

The Create Quote page opens.

2. In the Quote Name field, enter either the full Customer Name or a partial value and click Go to search for and select a customer.
3. Optionally:
   a. Enter a Contact Name or enter a partial value and click Go to search for the contact.
   b. Change the defaulted quote-to address, if any, or click Go to search for customer addresses.
   c. Change the default expiration date.
   d. Select a different Primary Salesperson.

Note: The Primary Salesperson defaults to the user creating the quote, if that user is set up as a salesperson for the current operating unit. If not, it defaults to the sales representative designated in the profile option **ASO: Default Salesrep**.

   e. From the menu, choose a Primary Sales Group.
   f. From the menu, choose a Sales Channel.
   g. Select a source for the quote.

      The source name refers to marketing sources that you set up in Oracle Marketing Online. If you create a quote from an opportunity, the source name defaults from Oracle Sales Online (OSO). If you are creating a quote directly, you can specify a source name in this field.
   h. From the menu, choose a different Order Type.
   i. From the menu, choose an Agreement.

      The Agreement menu lists only agreements that are valid for the customer.
   j. From the menu, choose a different Currency.

k. From the menu, choose a different Price List.

l. From the menu, choose a different Contract Template.

4. Click **Apply**.

   Processing creates the quote, and the Lines page opens. On the Lines page, you can add products to your quote. For more information, see Section 14.3, "Add Products to the Quote" on page 14-4.

14.3 Add Products to the Quote

The following steps describe how to add products to a quote.

Module

Oracle Sales Online

Navigation

Quotes tab > Lines page

Steps

1. Click the Add Product button.

   The Search and Select: Product page opens.

2. Verify that the Source is **Product Catalog**.

3. In the Product list, choose the Product Inventory category.

   Select **All** to search across all inventory categories.

4. In the text field, enter the part number or product name for an exact match, or partial information and % for a restricted or filtered list.

5. Click Go.

   The Product page shows the products matching your search criteria, including part number, product description, and unit of measure.

6. To add to the quote, select the Select check box next to the product(s).

7. Optionally, select a Line Price List for the selected product(s).

8. Click **Select** to add the product(s) to the quote and return to the Lines page, or choose **Select and Add More** to add the product(s) and search for additional products in different categories.

Tip: If a product is included (contained) in a Container Model, then you must search by the product name of the container. If you search by a product name that is in the container, then you cannot put those product in the quote.

On the Lines page, a line can have either:

- A circle with an **X**, which indicates that the product is configurable, but that the configuration is incomplete or invalid. To create a valid configuration, launch Oracle Configurator.
- A circle with a check mark, which indicates that the configuration is valid.

The lines can have a triangle icon that contains either a plus or a minus. This icon allows you to expand the view of a configured model:

- The plus indicates that the model view is in a collapsed state.
- The minus signs indicates that the model view is in an expanded state. You can view all components of a fully expanded model.

14.4 Reconfigure an Existing Service

The following procedures describe how to modify or change your customer's existing service.

Module

Oracle Sales Online

Navigation

Quotes tab > Quote Information > Lines page

Steps

1. Create a quote.

   For more information, see Section 14.2, "Create a New Quote" on page 14-2.

2. Add products to the quote from Install Base.

   For more information, see Section 14.5, "Add Products to Quote from Install Base" on page 14-6.

3. Search by Product Name, not by Container Name, in the Install Base for the telecommunications service(s) to reconfigure.
4. Reconfigure telecommunication service(s) in Oracle Configurator.
5. Place the order.

> **Note:** You cannot perform the following actions on a telecommunication service from Install Base: split line, remove changed lines, or add services.

14.5 Add Products to Quote from Install Base

Module

Oracle Sales Online

Navigation

Quotes tab > Quote Information > Lines page

Prerequisites

Display the quote to which you are adding products from Install Base.

Steps

1. Click the Add Products button.

   The Search and Select: Product page opens.
2. From the Source list, choose Installed Base.

   The Search and Select: Product page refreshes.
3. In the Search for Products field, choose Reconfigure.
4. Enter search criteria.

   > **Note:** You can search for product by Extended Attribute values, such as Attribute Name or Attribute Value.

5. Click Go.

   Search results appear.

6. In the Select column, select one or more products.
7. To add products from Install Base for reconfiguration, click the Select button.

   The installed service that you selected from Install Base appears in the appropriate container in the quote line. You can expanded the view of the container to show the selected product from Install Base.

14.6 Add to Container Model from Install Base

A search of a customer's Install Base could result in multiple search result pages. Currently, you can select only installed products from one search results page into a quote. To select products from additional search result pages you can:

- Use the **Add to Model from Installed Base** action on the quote lines pages. For more information, see the Steps section of this topic.
- Increase the number of rows appearing on a search result page. For more information, see Section 12.5, "Change Default Number of Rows" on page 12-15.

The following procedure describes how to add more items from Install Base to the container when you have a Container Model in a quote with items from Install Base. In the following procedure, your search for items is limited to items in the selected Container Model.

Module

Oracle Sales Online

Prerequisites

Display a quote with a container in the quote line.

Navigation

Quotes tab > Quote Information > Lines page

Steps

1. Select the container to which you want to add a product.
2. From the Action menu, choose **Add to Model From Installed Base**.
3. Click Go.

   The Search and Select: Installed Base Product page opens.
4. Specify your search criteria, and click Go.

Search results appear. The search is limited to items in the selected Container Model that you indicated in the first step of this procedure.

5. In the Select column, select the item to add to the quote's Container Model.
6. Click the Select button.

   The items that you selected appear in the container on the quote.

14.7 Remove Unchanged Components

An unchanged component is one that either:

- The user explicitly selected from Install Base, brought the component into a configuration session, but did not change it, or
- A passive component that was made active during the configuration session, but did not change.

If after reconfiguring the telecommunications service, you want to view only the changes, you can choose to remove all unchanged components. This is helpful to see the difference in cost after reconfiguration. Each unchanged component has a visual indicator in the Actions column, alerting you that the component is unchanged during the configuration session.

> **Note:** After you remove unchanged components in the quote, you cannot add the unchanged components to the quote by re-launching configurator. An unchanged component, when removed from the quote, continues to be visible in Configurator. The visual indicator is a triangle that contains an exclamation point.

Module

Oracle Sales Online

Navigation

Quotes tab > Quote Information > Lines page

Prerequisites

The line must be unchanged from the original configuration.

Important: You cannot undo the Remove Unchanged Components action. Before performing this action, ensure that this is the action that you want to perform.

Step

1. Click **Remove Unchanged Products**.

   This process removes all unchanged items in any configuration on the quote. Processing removes unchanged components, and the configuration view collapses. Expand the levels to view the telecommunications service. The pricing for the telecommunication service reflects only the changed components.

14.8 Remove Lines

You can remove unchanged components individually or remove entire Container Models.

Module

Oracle Sales Online

Navigation

Quotes tab > Quote Information > Lines page

Prerequisites

You have searched for and are displaying the quote that contains the line to remove.

The line to remove must not be a changed component of the Container Model.

Steps

1. Select the line to remove.
2. From the Actions menu, choose **Remove Line**.
3. Click Go.

   The line no longer appears on the Lines page.

14.9 Place an Order

When you convert the quote into an order, Oracle Service Fulfillment Manager (SFM) retrieves the order from Oracle Order Management. SFM fulfills the order and the service changes appear in Install Base.

The following steps describe how to place an order by converting a quote to a sales order.

Module

Oracle Sales Online

Navigation

Quotes tab > Overview link > Quote Overview page

Prerequisites

The order must be in Booked status.

There is a valid transition defined from the current status to Order Submitted.

You must have Update access to the quote.

The quote is in the highest version.

At least one line must be in the quote.

The quote is has not expired.

Step

1. From the Actions menu, choose **Place Order**.

   The Overview page refreshes and displays confirmation that the quote was submitted as an order. The Order Number appears as a link.

15

Administering the Solution

Topics in this chapter include:

- Section 15.1, "Re-validate Installed Configurations After Modifications" on page 15-2
- Section 15.2, "Managing Notification Errors" on page 15-3
- Section 15.3, "Find an Order in the SFM Order Flow-Through Area" on page 15-4
- Section 15.4, "Retry Install Base Updates" on page 15-4
- Section 15.5, "Retry Failed Outgoing Messages" on page 15-5
- Section 15.6, "Manage Failure Notifications in HTML" on page 15-7

See Also

For non-administrative end-user tasks, see Chapter 14, "Using the Solution".

15.1 Re-validate Installed Configurations After Modifications

If you modify a Container Model in either Oracle Applications or Oracle Configurator Developer, model data can become out of synchronization with configured components at a customer's site. If you make changes to a Container Model in either Oracle Applications or Oracle Configurator Developer, it is strongly recommended that you re-validate any installed configurations.

The following are examples of changes that occur in Oracle Applications to a Container Model that require you to re-validate installed configurations:

- Deleting a trackable child Model.
- Making a trackable child Model non-trackable.

The following are examples of changes made in Oracle Configurator Developer that require you to re-validate a Container Model:

- Deleting a Connector node from a trackable model when a rule contains participants from the Connector's parent and the target model.
- Adding, deleting, or changing a rule defined using a Connector, when the rule contains participants from two trackable components.
- Adding, deleting, or changing a rule in a trackable model that is not defined using a Connector, but the rule propagates to a participant in a rule defined using a Connector between trackable components.
- Adding, deleting, or changing any item that is a participant in a rule defined using a Connector, when the Connector's parent and target are both trackable.

Module

Oracle Configurator Developer

Responsibility

Configurator Administrator or Configurator Developer

Steps

1. Open the Container Model.
2. Regenerate the Active Model.
3. Republish the model.
4. In Oracle Quoting, create a new quote.

5. Search for all components within the customer's existing network configuration.
6. Select all of the components, and add them to the quote.

   If this step fails due to a large number of components, create multiple quotes with fewer sets of connected components, then continue the validation process separately for each quote. For each quote, try to select sets that have very few, if any, connections between each set.
7. If the changes made to the Container Model do not affect the customer's installed configurations, verify that all items appear with your **No Action** or **Reprice** Line Type.

   Because you define Line Types in Oracle Order Management, the exact text can vary. For more information, see the *Oracle Order Management Suite Implementation Manual*.
8. Verify that any items affected by the changes made in the Container Model are invalid within the configuration.
9. Submit the quote.

   If no errors occur, you have re-validated the configuration successfully.
10. If errors occur:
    a. Launch Oracle Configurator.
    b. Update the configuration to resolve each error.
    c. Resubmit the quote.

       Repeat steps a-c until you can submit the quote with no errors.

15.2 Managing Notification Errors

Enhancements to Service Fulfillment Manager (SFM) that assist you with notification errors while using the Oracle Telecommunications Service Ordering solution include:

- Incorporating enhancements to allow you to act upon notifications created as a result of a failed action.
- Improving interactions with third parties by automatically re-sending XML messages to them if the original XML message failed to send.

Notification error-related tasks include:

- Section 15.4, "Retry Install Base Updates" on page 15-4
- Section 15.5, "Retry Failed Outgoing Messages" on page 15-5
- Section 15.6, "Manage Failure Notifications in HTML" on page 15-7

15.3 Find an Order in the SFM Order Flow-Through Area

You should monitor your sales orders to check for error notifications.

Module

Oracle Applications (Forms)

Responsibility

SFM System Administrator

Navigation

Operations > Flow-Through Manager

Steps

1. In Order Num field:
   - Specify the OM Order Number.
   - Choose Sales Order as Source Application in the Search window.
   - Click Go.
2. Expand the order in the tree to view the Service Fulfillment lines that Service Fulfillment Manager has received.
3. Choose a line and navigate to the Work Items tab.
4. Click Parameters.
5. View the parameters that the system passed from the line into Service Fulfillment Manager.

15.4 Retry Install Base Updates

Service Fulfillment Manager (SFM) updates the Install Base with provisioning data after an order is complete in SFM. If the Install Base update fails, SFM sends a

response notification to the SFM System Administrator. The System Administrator has the option to retry the failed action.

It is recommended that you routinely monitor sales orders to be aware of errors that can occur, such as when provisioning fails. Provisioning failure can occur when there is an incompatible discrepancy between the order and the information in either Service Fulfillment Manager or Install Base. A provisioning failure appears on the Sales Order form in the status of the flow.

After you have fixed any problems pertaining to the incompatible discrepancy, you can retry to update the Install Base.

Module

Oracle Applications (Forms)

Responsibility

SFM System Administrator

Navigation

Operations > Flow Through Manager

Steps

1. Enter search criteria to find the Order Number.
2. Select the Order Number, and click View Details.

   The Order Information window appears.
3. Click the Notification button.
4. The Notification Inbox opens.
5. Select a notification whose status is Open.
6. Click View Details.

   For more information on your options, such as retrying to update the Install Base, refer to the *Oracle Service Fulfillment Manager Implementation Guide*.

15.5 Retry Failed Outgoing Messages

The XDP Resubmit Failed Message concurrent program resends XML messages that have failed to be sent successfully to 3rd party systems. The concurrent program takes a message code as input. When you run the concurrent program, SFM

identifies and attempts to resend all failed messages with the specified message code.

You should run the XDP Resubmit Failed Message program if:

- You have information that some of your messages failed to send
- You want to monitor and verify the sending of your messages.

Module

Oracle Applications (Forms)

Responsibility

SFM System Administrator

Navigation

Concurrent > Run Requests

Prerequisites

You need to know the Message Code of the message that you want to check.

Steps

1. Select Single Request and click OK.
2. In the Name field, choose XDP Resubmit Failed Message.
3. Click OK.
4. Enter the Message Code.
5. Click OK.

   Any messages that the system had failed to send reappear in the Outbound Message Queue. An adapter program polls the queue to send messages from the queue.
6. To view messages in the Outbound Message Queue, choose Administration > Queue Console.
7. In the Queue Console, select and open the Outbound Message Queue.

   The Entries field shows the number of messages in the queue.
8. Select a message and click the View Details button.

15.6 Manage Failure Notifications in HTML

SFM includes an HTML version of the Fallout Manager that you access to manage failure notifications. You view the automatically created failure notifications in HTML using Oracle Workflow. From the Workflow Notification page, you can navigate to the SFM HTML user interface to modify the work item parameters and retry the failed action.

Module

Oracle Applications (Forms)

Responsibility

Workflow Administrator Web (New)

Navigation

Status Monitor area

Prerequisites

Section 13.3, "Add Function for Menu Notifications" on page 13-5

Steps

1. In the Status Monitor area, click the Notifications link.
2. The Workflow Notifications page opens and displays: Type, Subject, and notification Date.
3. To view details of the order number notification, click the Subject hypertext link.

   The Notification Details page shows the selected failure notification message and hypertext links that let you fix the errors, such as Workitem Parameters.
4. In the Details area, click the link to update Workitem Parameters.

   The Workitem Parameters for Failed Order Line page opens.
5. In the Retry Value column, enter corrections to the workitem parameters.
6. Click the Update button.

   The Workitem Parameters for Failed Order Line page closes and the Notification Message Details page opens.
7. Click the Retry button.

Part IV

Appendixes

This part contains the following appendices:

- Appendix A, "Assumptions and Restrictions"
- Appendix B, "Functional Companion Code Examples"

A

Assumptions and Restrictions

A.1 List of Assumptions and Restrictions

The following is a list of assumptions and restrictions for the current release of Oracle Telecommunications Service Ordering (TSO) functionality:

- To successfully reconfigure an instance, you must accurately follow all setup steps and tips. For more information, see Part II, "Functionality and Setup".
- To reconfigure instances in Install Base, you must define and house models in a Container Model.
- In this release, only items that you can provision are in scope. Shippable and manufacturable items are out of scope. Models must be PTO. Non-ship models must be complete.
- Users do not add quantities greater than 1 for serialized, Install Base-trackable leaf items.
- Users do not connect two or more Connectors from one source item to one target item.
- It is recommended that users do **not** use pricing on Oracle Configurator Summary window. It is recommended that all pricing of telecommunications products should take place through Oracle Quoting. Oracle Quoting can use all the functionality that Advanced Pricing provides, which includes pricing based on attributes and sourcing rules.
- Internal representatives do not reconfigure configurations from Order Management
- There are no changes to pending orders. Current release designs do not support either:
  - Placing new quotes based on pending orders, or

  - Modification of entered but non-booked orders by launching Configurator from the Order Management Sales Order Pad.

- During fulfillment, Service Fulfillment Manager can only change user attribute values. No changes occur to structure, quantity, or connection information.
- Configurator cannot infer disassembly sequences from the delta computations. Consulting must build the fulfillment workflows.
- Reconfiguration can only be quoted to the same sold to customer that owns the service in Install Base. For example, Customer X cannot reconfigure services belonging to Customer Y.
- Fulfillment of orders occur in the same order in which they were booked.
- Certain types of **substantial** changes to a configuration model require the developer to re-validate existing instances. These changes include:
  - Changing a trackable multiple instantiable child Model of the Container model to non-trackable.
  - Deleting a trackable multiple instantiable child Model of the Container model.
  - Deleting a Connector node from one trackable model if there are connection rules with participants across the Connector.
  - Adding, deleting, or changing a connection rule between trackable items.
  - Adding, deleting, or changing any item that is a participant of a connection rule between trackable items.

B

Functional Companion Code Examples

Main sections in this appendix include:

- Section B.1, "Setting the Default Location" on page B-3
- Section B.2, "Setting the Line Type" on page B-5
- Section B.3, "Using Configuration Attributes with Install Base (IB)" on page B-12
- Section B.4, "Changing the Instance Name" on page B-22
- Section B.5, "Determining Editability" on page B-24

About this Appendix

This appendix contains code examples that support other chapters of this document. These examples are fuller and longer than the examples provided in the rest of this document, which are often fragments. See the cited background sections for details.

> **Note:** For examples that support other features of Oracle Configurator, refer to the *Oracle Configurator Implementation Guide* and the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

Table B–1 Code Examples Provided

Purpose of Example	Example
Section B.1, "Setting the Default Location"	Example B–1, "Setting the Default Location (DefaultLocation.java)"
Section B.2, "Setting the Line Type"	Example B–2, "Setting the Line Type (LineTypeFunc.java)"
Section B.3, "Using Configuration Attributes with Install Base (IB)"	Example B–3, "Using Configuration Attributes with Install Base (CZIBAttributeFuncComp.java)" Example B–4, "Using Configuration Attributes with Install Base (OptionalAttributeComp.java)"
Section B.4, "Changing the Instance Name"	Example B–5, "Changing the Instance Name (ChangeInstanceName.java)"

You should consult these other documents for details on the tasks described in this section:

- For information on how to write and compile Functional Companions, and on how to incorporate them into your configuration model, see the *Oracle Configuration Interface Object (CIO) Developer's Guide*.
- For information on how to install Functional Companions, see the *Oracle Configurator Installation Guide*.
- For an explanation of updating configurations, see Chapter 2, "TSO Business Process".
- For an details on how to build a configuration model that enables you to update configurations, see Section 10.2, "Create the Configuration Model" on page 10-16.

B.1 Setting the Default Location

This example demonstrates how to use a Functional Companion to set the default Location ID for the components being configured.

- To use the example, you can modify it according to the instructions in Section 11.1.2, "Specify the Default Location" on page 11-5.
- For general information, see Chapter 11, "Set Up Configurator Functional Companions".

> **Note:** If you customize this Functional Companion, you must place it in a Java package of your own, and not in `Oracle.*`.

Example B–1 Setting the Default Location (DefaultLocation.java)

```
package oracle.apps.cz.fc;

import oracle.apps.cz.cio.AutoFunctionalCompanion;
import oracle.apps.cz.cio.LogicalException;
import oracle.apps.cz.cio.IRuntimeNode;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;
import oracle.apps.cz.utilities.CZUiUtilities;
import oracle.apps.fnd.common.VersionInfo;
import com.sun.java.util.collections.*;
import com.sun.java.util.collections.Iterator;

public class DefaultLocation extends AutoFunctionalCompanion
{

  static Integer m_defaultLocation;
  static boolean m_attempted = false;
  static String m_defaultLocationTypeCode = "HZ_PARTY_SITES";

  private static void initializeDefaultLocation(IRuntimeNode node) {
    try{
      Connection conn  = node.getConfiguration().getContext().getJDBCConnection();
      String fetch = "SELECT party_site_id " +
        "FROM apps.hz_party_sites " +
        "WHERE sysdate BETWEEN NVL(start_date_active, sysdate) and " +
```

```
        "NVL(end_date_active, sysdate) " +
        "AND ROWNUM < 2 " ;
      Statement pstmt =  conn.createStatement();
      try {
        m_attempted = true;
        ResultSet rs = pstmt.executeQuery(fetch);
        if (rs.next()) {
          m_defaultLocation = new Integer(rs.getInt(1));
        }
        rs.close();
      } finally {
        if (pstmt != null) pstmt.close();
      }
    } catch(Throwable ex) {
      throw new RuntimeException(CZUiUtilities.stackTraceToString(ex));
    }
  }

  public void onSave() throws LogicalException {
    IRuntimeNode node = getRuntimeNode();
    if (!node.isTrackableRoot()) {
      throw new RuntimeException("DefaultLocation FunctionalCompanion is only required for
TrackableRoots");
    }
    if (!m_attempted) {
      initializeDefaultLocation(node);
    }
    if (m_defaultLocation == null) {
      throw new RuntimeException("Unable to initialize a Default Location from APPS.HZ_PARTY_
SITES table.");
    }
    setLocation(node);
  }

  private void setLocation(IRuntimeNode node) {
    node.setLocationID(m_defaultLocation);
    node.setLocationTypeCode(m_defaultLocationTypeCode);
    for (Iterator itr = node.getSummaryChildren().iterator(); itr.hasNext(); ) {
      setLocation((IRuntimeNode)itr.next());
    }
  }
}
```

B.2 Setting the Line Type

This example demonstrates how to use a Functional Companion to set the Line Type (fulfillment action) for the components being configured.

- To use the example, you can modify it according to the instructions in Section 11.1.3, "Specify the Line Type" on page 11-9.
- For general information, see Chapter 11, "Set Up Configurator Functional Companions".

Example B–2 Setting the Line Type (LineTypeFunc.java)

```
//package oracle.apps.cz.test.attribute;

import oracle.apps.cz.cio.AutoFunctionalCompanion;
import oracle.apps.cz.cio.Configuration;
import oracle.apps.cz.cio.NoSuchChildException;
import oracle.apps.cz.cio.IRuntimeNode;
import oracle.apps.cz.cio.BomNode;
import oracle.apps.cz.cio.Connector;
import oracle.apps.cz.cio.BomModel;
import oracle.apps.cz.cio.RuntimeNode;
import oracle.apps.cz.cio.LogicalException;
import com.sun.java.util.collections.ArrayList;
import com.sun.java.util.collections.HashMap;
import com.sun.java.util.collections.Iterator;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;
import oracle.apps.cz.utilities.CZUiUtilities;
import oracle.apps.fnd.common.VersionInfo;
import com.sun.java.util.collections.*;

public final class LineTypeFunc extends AutoFunctionalCompanion
{

/**
  FulfillmentAction describes the steps of "How to perform the update" or
  "How to fulfill the order line from the order".
  The update is the change (delta) in the runtimeNode state. The change may involve
  logic state change, quantity change, target change, location change or deletion.
```

```
  The change is determined with respect to the baseline configuration and computed
  after every commit (actually every time CIO walks the item tree).

  Customers are expected to define their own custom FulfillmentActions in
  the ONT.OE_TRANSACTION_TYPES_TL table, where:
  TRANSACTION_TYPE_ID refers to FullfillmentActionID
  NAME refers to the name of the FullfillmentAction.
  DESCRIPTION refers to the detailed steps

  Customers will write a FunctionalCcompanion that will determine the nature of change
  and accordingly set the FulfillmentActionType on the runtime node (setFulfillmentAction(MOVE)).
*/

  // Users should map CZ changes to ONT.OE_TRANSACTION_TYPES_TL for FulfillmentAction.
  // Users should hard code the values here or set these values in a way
  // that will satisfy their requirements but make sure that a record exists
  //  in ONT.OE_TRANSACTION_TYPES_TL with relevant information.
  // This example uses a custom table: MACD_FULFILLMENT_STRUCTURES
  // JOIN MACD_FULFILLMENT_STRUCTURES and ONT.OE_TRANSACTION_TYPES_TL with
  // MACD_FULFILLMENT_STRUCTURES.TRANSACTION_TYPE_ID = ONT.OE_TRANSACTION_TYPES_TL.TRANSACTION_
TYPE_ID

 // CZ changes
  public static String ADD  = "ADD" ;
  public static String QUANTITY = "VALUE_CHANGE";
  public static String DELETE =  "DELETE";
  public static String TARGET_CHANGE = "TARGET_CHANGE";
  public static String ATTRIBUTE_CHANGE = "ATTRIBUTE_CHANGE";
  public static String LOCATION_CHANGE = "LOCATION_CHANGE";
  public static String NAME_CHANGE = "NAME_CHANGE";
  public static String CHILD_CHANGE = "CHILD_CHANGE";


  // data structure to hold the czFulfillmentRecords that are sorted by priorty
  // of the delta change. Example add has precedence over value change
  public ArrayList m_czFulfillmentRecords;

  static Map m_cachedMap =  (Map)Collections.synchronizedMap(new HashMap());


  private void initializeDataForInstance(BomNode node) {
  // initialize the data from custom table
    int inventoryItemId =  node.getInventoryItemId();
    int organizationId =    node.getOrganizationId();
```

```
    lineInfoForBom lineInfo = new lineInfoForBom(inventoryItemId, organizationId,
node.getConfiguration() );
    // check to see if it is already cached
    m_czFulfillmentRecords = (ArrayList)m_cachedMap.get(lineInfo);
    if (m_czFulfillmentRecords != null) return;
    try{
      Connection conn  = node.getConfiguration().getContext().getJDBCConnection();
      String fetch  = "SELECT ONT_TRANSACTION_TYPE_ID , " +
                      "CZ_DELTA_TYPE , " +
                      "PRIORITY "  +
                      "from CZ.MACD_FULFILLMENT_STRUCTURES " +
                      "where INVENTORY_ITEM_ID =  " +inventoryItemId +
                      " and ORGANIZATION_ID = " + organizationId+ " order by PRIORITY";
     Statement pstmt =  conn.createStatement();
     try {
       ResultSet rs = conn.createStatement().executeQuery(fetch);
       while (rs.next()) {
        if (m_czFulfillmentRecords == null) {
          m_czFulfillmentRecords = new ArrayList();
          m_cachedMap.put(lineInfo, m_czFulfillmentRecords);
        }
        m_czFulfillmentRecords.add(new CzFullfillmentRecord (rs.getString(2), rs.getInt(1),
rs.getInt(3)));
       }
       rs.close();
     } finally {
      pstmt.close();
      // conn.close();
     }
    } catch(Throwable ex) {
      m_czFulfillmentRecords = null;
      // write to a log here
      ex.printStackTrace();
      throw new RuntimeException(CZUiUtilities.stackTraceToString(ex));
    }
  }

  public LineTypeFunc()
  {
  }


  public void onSave() throws LogicalException {
  onSummary();
  }
```

```
  /**
  * This will determine the nature of change and set the fulfillment action
  * for the runtime nodes
  */
  public void onSummary() throws LogicalException {
   // walk the tree to determine and set the fulfillment action
   RuntimeNode node = ((RuntimeNode)getRuntimeNode());
   if (node instanceof BomNode ) {
     initializeDataForInstance((BomNode)node);
     if (m_czFulfillmentRecords == null) {
       throw new RuntimeException("FulfillmentAction data not initialized for " +
node.getName());
     }
     setFulfillmentAction(node);
   }
  }

  public void setFulfillmentAction(RuntimeNode node) {
    if (node.getLineType() != null) {
      // unset the line type
      node.setLineType(null);
    }
    if (node.isChanged()) {
       node.setLineType(new Integer(determineFulfillmentAction(node)));
    }
    for (Iterator itr = node.getSummaryChildren().iterator(); itr.hasNext(); ) {
      setFulfillmentAction((RuntimeNode)itr.next());
    }
  }

  /**
  * This API determines all the changes associated with the node and uses
  * the changes (combination of changes) to determine the fulfillmentAction.
  *
  * Any logic that influences the FulfillmentAction should be used here.
  * For example: if TotalDiscountPrice > 1000, give a free service when fulfilling
  * the action.
  */
  public int determineFulfillmentAction(RuntimeNode node) {
  // get all the possible changes
    boolean add = node.isAddChanged();
    boolean delete = node.isDeleteChanged();
    boolean value = node.isValueChanged();
    boolean target = node.isTargetChanged();
```

```
boolean attr = node.isAttributeChanged();
boolean location  = node.isLocationChanged();
boolean name =  node.isNameChanged();
boolean child =  node.isChildChanged();

/*
 The line type of MOVE is currently being set on the connector (which is the
  item that has undergone the target change).  However, the telecom solution has
  been specified such that a target change on a "Network Link" should result in
  a line type of MOVE for the parent of the connector as well.  A "Network Link"
  is defined as a Bom Model with no location and exactly 2 connector children.
  So, the line type companion needs to be changed in the following way:
  For each node being examined
    if it is a BomModel
      if it has no location
        if number of direct children that are connectors == 2
          if either of those connectors has isTargetChanged() == true
            line type = MOVE
 */
boolean networkLink = isNetworkLinkAndLocationChanged(node);

for (Iterator itr = m_czFulfillmentRecords.iterator(); itr.hasNext();) {
  CzFullfillmentRecord czRecord = (CzFullfillmentRecord)itr.next();
  int fulfillmentAction =  czRecord.getFulfillmentActionId();
  String delta = czRecord.getCZDelta();

  if (networkLink && delta.equalsIgnoreCase(TARGET_CHANGE) ) {
    return fulfillmentAction;
  }
  if (delta.equalsIgnoreCase(ADD) && add) {
    return fulfillmentAction;
  }
  if (delta.equalsIgnoreCase(DELETE) && delete) {
    return fulfillmentAction;
  }
  if (delta.equalsIgnoreCase(QUANTITY) && value) {
    return fulfillmentAction;
  }
  if (delta.equalsIgnoreCase(TARGET_CHANGE) && target) {
    return fulfillmentAction;
  }
  if (delta.equalsIgnoreCase(NAME_CHANGE) && name) {
    return fulfillmentAction;
  }
  if (delta.equalsIgnoreCase(ATTRIBUTE_CHANGE) && attr) {
```

```
        return fulfillmentAction;
      }
      if (delta.equalsIgnoreCase(CHILD_CHANGE) && child) {
        return fulfillmentAction;
      }
      if (delta.equalsIgnoreCase(LOCATION_CHANGE) && location) {
        return fulfillmentAction;
      }
    }
     BomNode tRoot = ((BomNode)getRuntimeNode());
    throw new RuntimeException("Unable to determine FulfillmentAction for "  + node.getName() +
                               ". InventoryItemId/OrganizationId of its Trackable Root " +
tRoot.getCaption() +
                                " is " + tRoot.getInventoryItemId() + "/" +
tRoot.getOrganizationId() +
                                ". Please initialize data for the trackable Instance in CZ.MACD_
FULFILLMENT_STRUCTURES ");
  }

  public static void main(String[] args) {
    //System.out.println(metaDataOK);
  }

  class CzFullfillmentRecord {
    int m_fulfillmentActionId;
    String m_czDelta;
    int m_priority;

    CzFullfillmentRecord () {
      m_fulfillmentActionId = -1;
      m_czDelta = null ;
      m_priority = -1;
    }

    CzFullfillmentRecord (String czDelta, int actionId, int priority) {
      this();
      m_fulfillmentActionId = actionId;
      m_czDelta =czDelta ;
      m_priority = priority;
    }

    public String getCZDelta() {
      return m_czDelta;
    }
```

```
    public int getFulfillmentActionId() {
      return m_fulfillmentActionId;
    }

    public int getPriority() {
      return m_priority;
    }
  }

  class lineInfoForBom {
    int m_inventoryItemId;
    int m_organizationId;
    Configuration m_config;

    lineInfoForBom(int inventoryItemId, int organizationId, Configuration config) {
       m_inventoryItemId =  inventoryItemId;
       m_organizationId =   organizationId;
       m_config = config;
    }

    public boolean equals (Object obj) {
      return  m_config == ((lineInfoForBom)obj).m_config &&
              m_inventoryItemId == ((lineInfoForBom)obj).m_inventoryItemId &&
              m_organizationId == ((lineInfoForBom)obj).m_organizationId;
    }

    public int hashCode () {
      int result = 17;
      result = 37 * result + m_inventoryItemId;
      result = 37 * result + m_organizationId;
      return result;
    }
  }

  /**
     A "Network Link" is defined as a Bom Model with no location and exactly 2 connector
children.
      So, the line type companion needs to be changed in the following way:
      For each node being examined
        if it is a BomModel
          if it has no location
            if number of direct children that are connectors == 2
              node is a NetworkLink
                if either of those connectors has isTargetChanged() == true
                  node is a NetworkLink and location changed
```

```
   */
  private boolean isNetworkLinkAndLocationChanged(RuntimeNode model) {
    if (model.getType() != RuntimeNode.BOM_MODEL) return false ;
    if (model.getLocationID() != null) return false;

    Collection connectors = model.getChildrenByType(RuntimeNode.CONNECTOR);
    if (connectors.size() != 2) {
      return false;
    }
    for (Iterator connItr = connectors.iterator(); connItr.hasNext();) {
      if (((Connector)connItr.next()).isTargetChanged()) {
        return true;
      }
    }
    return false;
  }
}
```

B.3 Using Configuration Attributes with Install Base (IB)

This example demonstrates how to use a Functional Companion to collect configuration attribute values for the components being configured. This example is specific to integration with Oracle Install Base.

- To use the example, you can modify it according to the instructions in Section 11.1.5.5, "Modify the IBAttribute Functional Companion" on page 11-23.
- For general information, see Chapter 11, "Set Up Configurator Functional Companions".

Example B–3 Using Configuration Attributes with Install Base (CZIBAttributeFuncComp.java)

```
import oracle.apps.cz.cio.Attribute;
import oracle.apps.cz.cio.AutoFunctionalCompanion;
import oracle.apps.cz.cio.Property;
import oracle.apps.cz.cio.OptionFeatureNode;
import oracle.apps.cz.cio.BooleanFeature;
import oracle.apps.cz.cio.Configuration;
import oracle.apps.cz.cio.CountFeature;
import oracle.apps.cz.cio.IntegerNode;
import oracle.apps.cz.cio.Component;
import oracle.apps.cz.cio.ITrackableInstance;
import oracle.apps.cz.cio.DecimalNode;
import oracle.apps.cz.cio.ComponentSet;
```

```
import oracle.apps.cz.cio.NoSuchChildException;
import oracle.apps.cz.cio.BomNode;
import oracle.apps.cz.cio.TextNode;
import oracle.apps.cz.cio.BomModel;
import oracle.apps.cz.cio.Option;
import oracle.apps.cz.cio.RuntimeNode;
import oracle.apps.cz.cio.IRuntimeNode;
import com.sun.java.util.collections.Map;
import com.sun.java.util.collections.Collection;
import com.sun.java.util.collections.HashMap;
import com.sun.java.util.collections.ArrayList;
import com.sun.java.util.collections.Iterator;
import com.sun.java.util.collections.TreeSet;
import com.sun.java.util.collections.List;
import java.util.StringTokenizer;
import oracle.apps.fnd.common.VersionInfo;

/**
* This Functional Companion is executed every time an instance is loaded.
* It has to be associated with the instance in Oracle Configurator Developer.
* The companion recursively traverses the instance tree to collect all the
* attributes and associate the attributes with the appropriate runtimeNode.
* When the configuration is saved, the CIO saves all the attributes to the database.
*

* See the Oracle CIO Developer's Guide for an overview of Functional Companions.
*

* See Oracle Configurator Methodologies for an overview of
* Configuration Attributes and how the Patchset H example Functional
* Companion is implemented.
* The documentation describes the structure of a sample BOM Model with
* Properties that will used to map and create node specfic attributes.
*

* This sample implementation uses the Attribute class defined in the oracle.apps.cz.cio package
* to create new Attribute objects, and writes the attribute data to the CZ_CONFIG_EXT_ATTRIBUTES
table.
*/

public class CZIBAttributeFuncComp extends AutoFunctionalCompanion
{

  public static final int ATTR_MODE_CURRENT = 0;
  public static final int ATTR_MODE_CURRENT_AND_IMMEDIATE_TRACKABLE_CHILDREN = 1;
```

```
public static final int ATTR_MODE_CURRENT_AND_ALL_TRACKABLE_CHILDREN = 2;

// map for holding attributes that needs to be propgated to optional instances under
// trackable root
// key is trackableRoot, value is Object[2]
// object[0] collection of attributes with mode 1
// object[1] collection of attributes with mode 2
public static final Map m_attributes = new HashMap();


public CZIBAttributeFuncComp()
{
}

/**
 * An onLoad() FC will be called while loading the instance.
 * This will collect all the attributes from trackable BOM nodes and write
 * them to the database when the configuration is saved.
 */
public void onLoad() {
  // walk the tree to collect attributes
  if (((RuntimeNode)getRuntimeNode()).isTrackableRoot()) {
    buildConfigAttributes((RuntimeNode)getRuntimeNode());
  }
}

/**
 * Walk the configuration tree to collect attributes.
 */
public void buildConfigAttributes(RuntimeNode node) {
  // do not collect attributes for ComponetSets
  if (node.getType() != IRuntimeNode.COMPONENT_SET) {
    getAttributes(node);
  }
  for (Iterator itr = node.getSummaryChildren().iterator(); itr.hasNext(); ) {
    buildConfigAttributes((RuntimeNode)itr.next());
  }
}
/**
 * Populates a map of attributes associated with this BOM node by
 * reading its properties from the runtimeNode. The map is keyed by
 * the attribute index and the values are Attribute info that will
 * be used to create the actual attribute objects.  */
private void getAttributes(RuntimeNode node) {
  int defaultMode = ATTR_MODE_CURRENT; // default
```

```
    // check to see if user provided a default mode for the node
    Property defaultProp=  node.getPropertyByName("ATTR_MODE");
    if (defaultProp != null) {
      defaultMode = defaultProp.getIntValue();
    }
    if (node.isIBTrackable()) {
      HashMap map = new HashMap();
      Collection coll = node.getProperties();
      for (Iterator itr = coll.iterator(); itr.hasNext();) {
        Property prop = (Property)itr.next();
        if (prop.getName().equals("ATTR_MODE")) {
          // value already set above
          continue;
        }
        if (prop.getName().startsWith("ATTR_")) {
        // Map gets filled in this API
          collectAttributeInfo(prop, node, map);
        }
      }
      // now get attribute info  and create attributes
      Collection values = map.values();
      for (Iterator itr = values.iterator(); itr.hasNext();) {
        int mode = -1;
        Object[] AttrInfo = (Object[])itr.next();
        String path = (String)AttrInfo[0]; // should not be null
        String attrName = null;
        String attrGroup = null;
        if (AttrInfo[1] != null) { // mode can be null
          mode = ((Integer)AttrInfo[1]).intValue();
        }
        if (AttrInfo[2] != null) { // name can be null
          attrName = (String)AttrInfo[2];
        }
        if (AttrInfo[3] != null) { // group can be null
          attrGroup = (String)AttrInfo[3];
        }
       createAndAssociateAttribute(path, attrName, attrGroup, (mode != -1 ? mode : defaultMode),
node);
      }
    }
  }

/**
Creates and associates an attribute between the BOM items and the attribute features.
*/
```

```
  private void createAndAssociateAttribute (String path, String attrName, String attrGroup, int
mode, RuntimeNode node) {
    if (mode != ATTR_MODE_CURRENT &&
        mode != ATTR_MODE_CURRENT_AND_IMMEDIATE_TRACKABLE_CHILDREN &&
        mode != ATTR_MODE_CURRENT_AND_ALL_TRACKABLE_CHILDREN ) {
      throw new RuntimeException("Unrecoginized attribute propagation mode \"" + mode + "\" for
node " + node.getName());
    }
    RuntimeNode valueFeature = findValueFeature(node, path);
    String[] attrInfo =  findAttrNameAndGroup(valueFeature);
    if (attrName == null || attrName.length() < 1) {
      attrName = attrInfo[0];
    }
    if (attrGroup== null || attrGroup.length() < 1) {
      attrGroup = attrInfo[1];
    }

    Attribute attribute = new Attribute (attrName, attrGroup, valueFeature);
    populateMapForOptionalInstances (node, mode, attribute);
    node.associateAttribute(attribute);
    propagateAttribute(node, attribute, mode, true);
  }

  /**
  * @params node oracle.apps.cz.cio.RuntimeNode
  * @params prop oracle.apps.cz.cio.Property
  * See the Oracle Configurator Developer User's Guide for an
  * overview of Properties.
  * For a Property, we will find the attribute name (ATTR_int_NAME), attribute group
  * (ATTR_int_GROUP), attribute mode (ATTR_int_MODE or ATTR_MODE).
  * Example:
  * Property name = ATTR_1_NAME
  * Property value = TestValue
  * We collect the info (for example, for attribute 1 the name is TestValue. This information
  * we store in the map (key is the attribute index and value is the info we collect).
  */
  private void collectAttributeInfo(Property prop, RuntimeNode node, HashMap map) {
    String name = prop.getName();
    String value = prop.getStringValue();
    // All attribute properties start with ATTR_
    // Get the index of this attribue
    Integer index = null;
    int beginningIndex = new String("ATTR_"). length();
    StringTokenizer tokens = new StringTokenizer(name.substring(beginningIndex), "_", false);
    if (tokens.hasMoreTokens()) {
```

```
    try {
      index = new Integer(tokens.nextToken());
    } catch (NumberFormatException nfe) {
      throw new RuntimeException("Attribute properties on BOM nodes should follow " +
                                 "the pattern ATTR_int_*");
    }
  }
  Object[] attrInfo = null;
  if (index != null) {
    attrInfo = (Object[]) map.get(index);
    if (attrInfo == null) {
      attrInfo = new Object[4];
      map.put(index, attrInfo );
    }
  } else {
    return;
  }

  // Assign the appropriate property to this attribute.
  if (name.endsWith("PATH")) {
    attrInfo[0] = prop.getStringValue();
  } else if (name.endsWith("MODE")) {
    attrInfo[1] = new Integer(prop.getIntValue());
  } else if (name.endsWith("NAME")) {
    attrInfo[2] = prop.getStringValue();
  } else if (name.endsWith("GROUP")) {
    attrInfo[3] = prop.getStringValue();
  } else {
    throw new RuntimeException("Attribute properties on BOM nodes should end up with " +
                               "with either \"PATH\", \"MODE\" ");
  }
}

/**
* Finds the default attribute name and attribute group. These will only be used if
* the Attribute (which has this valuefeature as its value) does not have a name or group.
*/
public String[] findAttrNameAndGroup(RuntimeNode valueFeature) {
  int names = 0;
  int groups = 0;
  String[] attrInfo = new String[2];
  for (Iterator itr = valueFeature.getProperties().iterator(); itr.hasNext();) {
    Property prop = (Property)itr.next();
    String value = prop.getStringValue();
    if (prop.getName().startsWith("ATTR_NAME")) {
```

```
        attrInfo[0] = value;
        names++;
      } else if (prop.getName().startsWith("ATTR_GROUP")) {
        attrInfo[1] = value;
        groups++;
      }
    }
    if (names > 1 || groups > 1 ) {
      throw new RuntimeException("Value feature has more then one attribute group or name");
    }
    return attrInfo;
  }

/**
* Propagate the attribute values all way down the tree based on the propagation mode the
attribute.
* (ATTR_MODE_CURRENT)0 = only where attached (default if omitted);
*  (ATTR_MODE_CURRENT_AND_IMMEDIATE_TRACKABLE_CHILDREN) 1 = where attached + immediate
TrackableBOM children;
* (ATTR_MODE_CURRENT_AND_ALL_TRACKABLE_CHILDREN) 2 = where attached + all Trackable BOM children
*/
  public void propagateAttribute(RuntimeNode node, Attribute attribute, int mode, boolean
nextLevel) {
    if (mode == ATTR_MODE_CURRENT) return;
    for (Iterator itr =  node.getSummaryChildren().iterator(); itr.hasNext(); ) {
      RuntimeNode childNode = (RuntimeNode)itr.next();
      if (childNode.isIBTrackable()) {
         // do not associate attributes for CompSets since they do not have a DbConfigItem.
        if ( childNode.getType() != RuntimeNode.COMPONENT_SET ) {
           // always overide the association. Assumption is that we propagate from top to bottom
of the tree.
          childNode.associateAttribute(attribute);
          populateMapForOptionalInstances (node, mode, attribute);
        } else if (nextLevel && mode == ATTR_MODE_CURRENT_AND_IMMEDIATE_TRACKABLE_CHILDREN) {
          // it is a compSet, we still to propagate to one level if mode = 1 to all immediate
children (ignore compSets)
          propagateAttribute(childNode, attribute, mode, false);
        }

        if ( mode == ATTR_MODE_CURRENT_AND_ALL_TRACKABLE_CHILDREN ) {
          // propagate the attribute values to all all Trackable BOM children
          propagateAttribute(childNode, attribute, mode, true);
        }
      }
    }
```

```
  }

 /**
 * Finds the valueFeature of the attribute with the given path. The path should start from the
 * nearest Bom Model. Example: IBCZRootNode.Component.Feature-1002
 * "." is StringTokenizer.
 */
  public RuntimeNode findValueFeature(RuntimeNode node, String path) {
    RuntimeNode valueFeature = findNearestBomModel(node);
    StringTokenizer tokenizer = new StringTokenizer(path, ".", false);
    while (tokenizer.hasMoreTokens()) {
      try {
        valueFeature = (RuntimeNode)valueFeature.getChildByName(tokenizer.nextToken(),
RuntimeNode.CURRENT_OR_DISCONTINUED_CHILD);
        if (valueFeature instanceof ComponentSet) {
          throw new RuntimeException("Cannot refer to a multiple instantiable Component: " +
                                     valueFeature.getName() + " in the Attribute Path");
        }
      } catch (NoSuchChildException nsce) {
        throw new RuntimeException("No attribute feature found in Path " +
                                   path + " for Node " + node.getName() );
      }
    }
    return valueFeature;
  }

  private BomModel findNearestBomModel(RuntimeNode node) {
    if (node.getType() == IRuntimeNode.BOM_MODEL) {
       return (BomModel)node;
    }
    IRuntimeNode parent = node.getParent();
    while (parent != null) {
      if (parent.getType() == IRuntimeNode.BOM_MODEL) {
        return (BomModel)parent;
      }
      parent = parent.getParent();
    }
    throw new RuntimeException(node.getName() + " does not have a ancestor Bom Model." );
  }

  public void populateMapForOptionalInstances (RuntimeNode node, int mode, Attribute attribute) {
    int nodeType = node.getType();
    if ((nodeType == RuntimeNode.BOM_MODEL || nodeType == RuntimeNode.BOM_OPTION_CLASS) &&
node.hasTrackableAncestor() ) {
      if (!m_attributes.containsKey(node)) {
```

```
        Object[] value =  new Object[2];
        value[0] = new ArrayList();
        value[1] = new ArrayList();
        m_attributes.put(node, value);
      }
      Object[] attributes = (Object[]) m_attributes.get(node) ;
      if (mode == ATTR_MODE_CURRENT_AND_IMMEDIATE_TRACKABLE_CHILDREN) {
        ((ArrayList)attributes[0]).add(attribute);
      } else if (mode == ATTR_MODE_CURRENT_AND_ALL_TRACKABLE_CHILDREN) {
        ((ArrayList)attributes[1]).add(attribute);;
      }
    }
  }

  public void terminate() {
    deleteNodesFromStaticMap(getRuntimeNode());
  }

  private void deleteNodesFromStaticMap(IRuntimeNode node) {
    m_attributes.remove(node);
    int nodeType = node.getType();
    if ((nodeType == RuntimeNode.BOM_MODEL || nodeType == RuntimeNode.BOM_OPTION_CLASS) &&
node.hasTrackableAncestor() ) {
      for (Iterator itr = node.getChildren().iterator(); itr.hasNext();) {
        deleteNodesFromStaticMap((IRuntimeNode)itr.next());
      }
    }
  }
}
```

Example B–4 Using Configuration Attributes with Install Base (OptionalAttributeComp.java)

```
//package oracle.apps.cz.test.attribute;

import oracle.apps.cz.cio.Attribute;
import oracle.apps.cz.cio.AutoFunctionalCompanion;
import oracle.apps.cz.cio.Property;
import oracle.apps.cz.cio.OptionFeatureNode;
import oracle.apps.cz.cio.BooleanFeature;
import oracle.apps.cz.cio.Configuration;
import oracle.apps.cz.cio.CountFeature;
import oracle.apps.cz.cio.IntegerNode;
import oracle.apps.cz.cio.ITrackableInstance;
import oracle.apps.cz.cio.DecimalNode;
```

```
import oracle.apps.cz.cio.Component;
import oracle.apps.cz.cio.NoSuchChildException;
import oracle.apps.cz.cio.BomNode;
import oracle.apps.cz.cio.TextNode;
import oracle.apps.cz.cio.BomModel;
import oracle.apps.cz.cio.Option;
import oracle.apps.cz.cio.RuntimeNode;
import oracle.apps.cz.cio.IRuntimeNode;
import com.sun.java.util.collections.Map;
import com.sun.java.util.collections.Collection;
import com.sun.java.util.collections.HashMap;
import com.sun.java.util.collections.ArrayList;
import com.sun.java.util.collections.Iterator;
import com.sun.java.util.collections.TreeSet;
import com.sun.java.util.collections.List;
import com.sun.java.util.collections.ListIterator;
import java.util.StringTokenizer;
import oracle.apps.fnd.common.VersionInfo;

public class OptionalAttributeComp extends CZIBAttributeFuncComp
{

  public OptionalAttributeComp() {
  }

  public void onLoad() {
    RuntimeNode node =  (RuntimeNode)getRuntimeNode();
    // walk the tree to collect attributes
    if (!node.hasTrackableAncestor() || node.isTrackableRoot()) throw new
RuntimeException("Invalid definition of onLoad() attribute companion.");
    if (node.isIBTrackable()) {
      List list = getParentNodes(node);
      int j = list.size();
      while (j <= list.size() && j != 0) {
        j--;
        Object[] attributes =   (Object[])m_attributes.get( list.get(j)) ;
        if (attributes != null) {
          propagateAttributesFromRoot((Collection)attributes[0], node, ATTR_MODE_CURRENT_AND_
IMMEDIATE_TRACKABLE_CHILDREN);
          propagateAttributesFromRoot((Collection)attributes[1], node, ATTR_MODE_CURRENT_AND_ALL_
TRACKABLE_CHILDREN);
        }
      }
      buildConfigAttributes(node);
    }
```

```
  }

  private List getParentNodes (RuntimeNode node) {
    ArrayList list = new ArrayList();
    Component root = node.getRootInstance();
    RuntimeNode parent = (RuntimeNode)node.getParent();
    while (true) {
      list.add(parent);
      if(parent == root) break;
      parent = (RuntimeNode)parent.getParent();
    }
    return list;
  }


  private void propagateAttributesFromRoot(Collection coll, RuntimeNode node, int mode) {
    for (Iterator itr = coll.iterator(); itr.hasNext();) {
      Attribute attribute = (Attribute) itr.next();
      node.associateAttribute(attribute);
      if (mode == ATTR_MODE_CURRENT_AND_ALL_TRACKABLE_CHILDREN) {
        propagateAttribute(node, attribute, mode, true);
      }
    }
  }
}
```

B.4 Changing the Instance Name

This example demonstrates how to use a Functional Companion to change the name of instances of the component being configured.

- To use the example, you can modify it according to the instructions in Section 11.1.4, "Change the Instance Name" on page 11-18.
- For general information, see Chapter 11, "Set Up Configurator Functional Companions".

Example B–5 Changing the Instance Name (ChangeInstanceName.java)

```
//Companion takes input from Text Feature and changes the corresponding instance name


import oracle.apps.fnd.common.VersionInfo;
import javax.servlet.http.HttpServletResponse;
```

```
import java.io.PrintWriter;
import java.io.*;
import oracle.apps.cz.cio.FunctionalCompanion;
import oracle.apps.cz.cio.AutoFunctionalCompanion;
import oracle.apps.cz.cio.IUserInterface;
import oracle.apps.cz.cio.*;
import oracle.apps.cz.cio.Component;
import com.sun.java.util.collections.List;
import com.sun.java.util.collections.Iterator;
import oracle.apps.cz.utilities.CheckedToUncheckedException ;
import com.sun.java.util.collections.ArrayList;

public class ChangeInstanceName extends FunctionalCompanion implements
IUserInterfaceEventListener{
  IUserInterface  mUi;
  Component rootnode;
  PrintWriter p = null;
  String textFeature = "InstanceName";

public void initialize(IRuntimeNode node, String name, String description, int id) {
        super.initialize(node, name, description, id);
        mUi = this.getRuntimeNode().getConfiguration().getUserInterface();
        if (mUi != null){
             mUi.addUserInterfaceEventListener(IUserInterfaceEvent.POST_VALUE_CHANGE, this);
       }
        rootnode = (Component)this.getRuntimeNode();
  }

  public void handleUserInterfaceEvent(IUserInterfaceEvent event) {
      if (event.getUiNode() != null ){
       try{
           if ((event.getUiNode().getType() == IUserInterfaceNode.TEXT_FIELD) && (
event.getUiNode().getName().equals(textFeature))){
              String inputText =
((TextFeature)event.getUiNode().getRuntimeNode()).getTextValue().toString();
              Component c  = (Component)mUi.getCurrentScreen().getRuntimeNode();
              c.setInstanceName(inputText);
           }
       }catch (Exception e){
           throw new RuntimeException("Error in handleUserInterfaceEvent");
       }
    }
  }
}
```

B.5 Determining Editability

This example demonstrates how to use a Functional Companion to determine whether a component in a network of configurable components is editable, when it is activated. This example uses the method IRuntimeNode.isEditable(), and the event type `IUserInterfaceEvent.POST_ACTIVATE_ACTION`.

For background, see Section 11.2.1.4, "Determining Editability" on page 11-28.

For general background, see Chapter 11, "Set Up Configurator Functional Companions".

Example B–6 Determining Editability (PostActivateAction.java)

```
// package oracle.apps.cz.cio.test;  // Use your own package

import com.sun.java.util.collections.ArrayList;
import com.sun.java.util.collections.Iterator;
import com.sun.java.util.collections.List;

import javax.servlet.http.HttpServletResponse;

import oracle.apps.cz.cio.AutoFunctionalCompanion;
import oracle.apps.cz.cio.BomOptionClass;
import oracle.apps.cz.cio.BomStdItem;
import oracle.apps.cz.cio.Component;
import oracle.apps.cz.cio.FunctionalCompanion;
import oracle.apps.cz.cio.IRuntimeNode;
import oracle.apps.cz.cio.IUserInterface;
import oracle.apps.cz.cio.IUserInterfaceEvent;
import oracle.apps.cz.cio.IUserInterfaceEventListener;
import oracle.apps.cz.utilities.CheckedToUncheckedException;

public class PostActivateAction extends FunctionalCompanion implements
IUserInterfaceEventListener{

  IUserInterface mUi;
  Component rootnode;

  public PostActivateAction()
  {
  }
```

```
// This method runs when the FC is initialized
public void initialize(IRuntimeNode node, String name, String description, int id) {
  super.initialize(node, name, description, id);

  // Get the UI object associated with the configuration
  mUi = this.getRuntimeNode().getConfiguration().getUserInterface();

  // Register the FC to listen for POST_ACTIVATE_ACTION events
  if (mUi != null){
    mUi.addUserInterfaceEventListener(IUserInterfaceEvent.POST_ACTIVATE_ACTION, this);
  }

  // Set the root node associated with this FC
  rootnode = (Component)this.getRuntimeNode();
}

// This method handles the event being listened for
public void handleUserInterfaceEvent(IUserInterfaceEvent event) {
  if (mUi == null){
    return;
  }
  if (event.getUiNode() != null ){
    try{
      String nodename = event.getUiNode().getName(); // The node triggering the event
      String message = nodename + " is now active!";
      String error = nodename + " is not activated";
      String title = "Test POST_ACTIVATE_ACTION";

      // Get the type of the node triggering the event
      int type = mUi.getCurrentScreen().getRuntimeNode().getType();
      if(type != IRuntimeNode.BOM_OPTION_CLASS) {
        Component c = (Component)mUi.getCurrentScreen().getRuntimeNode();

        // If the component is editable, show "activated" message
        if (c.isEditable()){
          mUi.addMessageBox(1,message,title);
        }
        else
          {
            // If the component is not editable, show "not-activated" message
            mUi.addMessageBox(1,error,title);
          }
      }
      // Make updates to the configuration here. This update is arbitrary.
      BomOptionClass oc1 = (BomOptionClass)rootnode.getChildByName("MY-PTO-OC1-T");
```

```
        BomStdItem item1 = (BomStdItem)oc1.getChildByName("MY-PTO-ITEM1-T");
        item1.setCount(10);
      } catch (Exception e) {
        throw new CheckedToUncheckedException(e);
      }
    }
  }
}
```

Glossary

This glossary contains definitions that you may need while implementing a Telecommunications Services Ordering solution. For terms specific to other Oracle Applications, refer to that application's documentation.

accepted

A quote status indicating that the customer has accepted the quote terms.

API

Application Programming Interface

ATO

Assemble to Order. An environment where you open a final assembly order to assemble items that customers order. Assemble-to-order is also an item attribute that you can apply to standard, model, and Option Class items. An item you make in response to a customer order.

baseline configuration

The existing configuration, used as the basis for computing a new configuration. Generally, the baseline configuration is the latest booked, provisioned, or installed configuration revision. When updating a configured item, you must update from this baseline. You can update the baseline to meet your new needs.

bill of material

A list of component items associated with a parent item and information about how each item relates to the parent item. Oracle Manufacturing supports standard, model, Option Class, and planning bills. The item information on a bill depends on the item type and bill type. The most common type of bill is a standard bill of material. A standard bill of material lists the components associated with a product

or subassembly. It specifies the required quantity for each component plus other information to control work in process, material planning, and other Oracle Manufacturing functions. Also known as product structures.

BOM

See bill of material.

BOM item

The node imported into Oracle Configurator Developer that corresponds to an Oracle Bills of Material item. Can be a BOM Model, BOM Option Class node, or BOM Standard Item node.

BOM Model

A model that you import from Oracle Bills of Material into Oracle Configurator Developer. When you import a BOM Model, effective dates, ATO rules, and other data are also imported into Configurator Developer. In Configurator Developer, you can extend the structure of the BOM Model, but you cannot modify the BOM Model itself or any of its attributes.

BOM Model node

The imported node in Oracle Configurator Developer that corresponds to a BOM Model created in Oracle Bills of Material.

BOM Option Class node

The imported node in Oracle Configurator Developer that corresponds to a BOM Option Class created in Oracle Bills of Material.

BOM Standard Item node

The imported node in Oracle Configurator Developer that corresponds to a BOM Standard Item created in Oracle Bills of Material.

category

Code used to group items with similar characteristics, such as plastics, metals, or glass items.

CIO

See Oracle Configuration Interface Object (CIO).

component item

An item associated with a parent item on a bill of material.

concurrent manager

Components of your applications concurrent processing facility that monitor and run time–consuming tasks for you without tying up your terminal. Whenever you submit a request, such as running a report, a concurrent manager does the work for you, letting you perform many tasks simultaneously.

concurrent process

A task in the process of completing. Each time you submit a task, you create a new concurrent process. A concurrent process runs simultaneously with other concurrent processes (and other activities on your computer) to help you complete multiple tasks at once with no interruptions to your terminal.

concurrent processing facility

An Oracle Applications facility that runs time-consuming, non-interactive tasks in the background.

concurrent program

Executable code (usually written in SQL*Plus or Pro*C) that performs the function(s) of a requested task. Concurrent programs are stored procedures that perform actions such as generating reports and copying data to and from a database.

concurrent request

A user-initiated request issued to the concurrent processing facility to submit a non-interactive task, such as running a report.

configurable item

A base model item that a user can configure by adding components.

configuration

A specific set of specifications for a product, resulting from selections that a user made in Oracle Configurator. *See also* partial configuration.

configuration attribute

A characteristic of an item that is defined in the host application (outside of its inventory of items), in the Model, or captured during a configuration session. Configuration attributes are inputs from or outputs to the host application at initialization and termination of the configuration session, respectively.

Configuration Interface Object

See Oracle Configuration Interface Object (CIO).

configuration model

Represents all possible configurations of the available options, and consists of model structure and rules. It also commonly includes User Interface definitions and Functional Companions. A configuration model is usually accessed in a runtime Oracle Configurator window. *See also* model.

configuration session

The time from launching or invoking to exiting Oracle Configurator, during which end users make selections to configure an orderable product. A configuration session is limited to one configuration model that is loaded when the session is initialized.

configurator

The part of an application that provides custom configuration capabilities. Commonly, a window that can be launched from a hosting application so end users can make selections resulting in valid configurations.

connected-to relationships

Models network connections and shows the service configuration. You can view the connected-to relationships of a configured instance from Oracle Install Base. If you define items in Oracle Inventory as link items, you can then use Oracle Install Base to show the start and end locations of the link. The locations for the link instance are the geographic addresses of the instance items.

Connector

The node in the model structure that enables an end user at runtime to connect the Connector node's parent to a referenced Model.

Container Model

A pick-to-order bill of material (BOM) model that you import from Oracle Bills of Material into Oracle Configurator Developer which supports multiple instantiation reconfiguration and active and passive components from Install Base.

CTO

Configure to Order

delta

The difference between the new and the baseline configuration.

drafted

A quote status indicating that the quote is in the initial phase.

entered

A quote status indicating that the quote has been successfully submitted as an order in Order Management. Orders with this status can be modified in Order Management.

fulfillment

Fulfilled sales order lines have successfully completed all Workflow processing activities up to the point of becoming eligible for invoicing.

Functional Companion

An extension to the configuration model beyond what you can implement in Configurator Developer.

An object associated with a Component that supplies methods that you can use to initialize, validate, and generate customer-centric views and outputs for the configuration.

ICX

Inter-Cartridge Exchange

Install Base

The sum total of all products that a company has responsibility to provide service for at customer sites.

instance

A single representation of an item in a configuration. *See also* instantiate. *Compare* count.

Also, the memory and processes of a database.

instance name

A method of capturing additional information about a product or instance. An instance name can be either automatically generated or manually assigned during a configuration session.

instantiate

To create an instance of something. Commonly, to create an instance of a component in the runtime user interface of a configuration model. *See also* multiple instantiation.

item

Anything you make, purchase, or sell, including components, subassemblies, finished products, or supplies. Oracle Manufacturing also uses items to represent planning items that you can forecast, standard lines that you can include on invoices, and Option Classes you can use to group options in model and Option Class bills.

item attributes

Specific characteristics of an item, such as order cost, item status, revision control, account, and so on.

location

A point in geographical space that a street address describes. Also, a shorthand name for an address. Location appears in address lists of values to let you choose the correct address based on an intuitive name. For example, you can specify the location name of **Receiving Dock** to the **Ship To** business purpose of 100 Main Street.

LOV

A list of values in a text field, from which the user must choose.

MACD

An acronym for the following actions: move, add, change, and disconnect. The Oracle Telecommunications Service Ordering solution enables these actions for a customer's telecommunications services.

multiple instantiation

The ability to configure a child Model or component of a top-level model multiple times during the same configuration session. The child Model appears in the bill of materials only once, but you can individually configure it as many times as needed while running Oracle Configurator.

network model

Refer to the Container Model definition.

notifications (notification id)

Notifications are instances of messages which some role receives. The message includes a row that shows status flags to record the state of the notification, date fields for when the notification was sent, due, and responded to. A new row appears in the Notifications table each time a role receives a message. The row persists even after the notification has been responded to, until a purge operation moves to close notifications to an archive.

one-time charge

A charge that a customer pays only once. Examples include installation fees, activation fees, and change fees.

Oracle Configuration Interface Object (CIO)

A server in the runtime application that creates and manages the interface between the client and the underlying representation of model structure and rules in the Term.

The CIO is the API that supports creating and navigating the Model, querying and modifying selection states, and saving and restoring configurations.

partial configuration

The ability to modify part of an existing configuration without launching the entire configuration in Oracle Configurator.

pick-to-order (PTO)

A configure-to-order environment where the options and included items in a model appear on pick slips and order pickers gather the options when they ship the order. Alternative to manufacturing the parent item on a work order and shipping it. Pick-to-order is also an item attribute that you can apply to standard, model, and Option Class items.

pick-to-order (PTO) item

A previously defined configuration order that pickers gather as separately finished included items just before they ship the order.

pick-to-order (PTO) model

An item with an associated bill of material with optional and included items. At order entry, the configurator is used to choose the optional items to include for the order. The order picker gets a detailed list of the chosen options and included items to gather as separately finished items just before order shipment.

provisioning or provisionable

When Oracle Order Management passes the sales order information on to Service Fulfillment Manager for provisioning. Provisioning involves capturing an order request, validating the order, analyzing the order, fulfilling the order, completing the order, and if necessary, managing order fallout. When you provision the order, Oracle Install Base receives the updated item information and the sales order is complete.

quote

A collection of items with pricing that a sales representative with a Sales Representative role creates on behalf of a customer in Oracle Quoting.

quote status

The quote status indicates the stage of preparation that a quote is in. Possible quote statuses include drafted, bid, accepted, entered, ordered, order problem, order reviewed, lost, and inactive.

reconfigure

Make changes to a configured model that a customer has already purchased.

recurring charge

A charge that the customer pays periodically (such as per month, quarter, or year) like a subscription fee.

SFM

Service Fulfillment Manager (SFM) provisions the telecommunication services that the customer ordered. *See also* provisioning or provisionable.

TSO

An acronym for the Oracle Telecommunications Service Ordering solution.

WIP (Work In Progress**)**

An item in various phases of production in a manufacturing plant. This includes raw material awaiting processing up to final assemblies ready to be received into inventory.

Workflow

This determines the header flow for an order transaction type or line flows possible for a line transaction type. There can be only one header flow associated with an order transaction type but a line transaction type can be coupled with different

order types and item types and there can be different flow couplings for the permitted transaction type and item type combinations.

Index

A

access
  discontinued items, 11-27
  instances, 11-27
access level, change Workflow, 13-4
action
  Activate Instance, 10-17
  add to Model from Installed Base, 14-7
  disable Add Service, 12-14
  disable Split Line, 12-14
  enable Add to Model Item From Install Base, 12-14
  enable Reconfigure, 12-14
  enable Remove Product, 12-14
  failed, 15-3, 15-5, 15-7
  for node, determine appropriate, 11-16
  for node, determine correct, 11-17
  fulfillment, 11-10, 11-17, B-5
  fulfillment based on, 3-10
  identify based on, 3-10
  known as Line Type, 3-3
  Line Type, 3-9
  Line Type, default, 5-4
  map work item to item and, 13-10
  menu, 12-2
  pricing dependency on Line Type or, 10-19
  provisioning, 2-8
  provisioning action in column, 3-4
  set up Lines page, 12-6
  specify from array of change types, 11-16
  suggested renaming of column, 12-7
Activate Instance action, 10-17
activation sequence of lines, about, 13-11
Active Model
  error appears when generating, 6-7, 10-7, 10-11, 10-14, 10-15
  generate, 10-16
  no error when generating, 10-12
  regenerate, 15-2
Add Service, disable action, 12-14
add service, from Install Base, 14-6
Add to Model from Installed Base, about, 14-7
Add to Model Item From Install Base, enable, 12-14
additional Line Type
  Quoting setup, 12-3
  reconfiguration setup, 12-3
address, specify for quote, 14-2
administration tasks, 15-1
Advanced Pricing
  set up, 9-1
agreement, choose for quote, 14-3
API, Oracle Configuration Interface Object, 11-25
approval, submit quote for, 2-10
ASO Automatic Numbering for quote, 14-3
ASO Default Salesrep profile option, 14-3
assumptions and restrictions, A-1
ATO BOM Models, 11-4
attribute
  configuration code example, B-12
  configuration, using with IB, 11-21
  create extended, 8-3
  extended. *See also* extended attribute.
  global level, 8-3
  instance level, 8-4
  inventory item level, 8-3
  item category level, 8-3

mandatory, 13-11
map column, 12-8
map extended, to SFM items, 8-6
persistence of values, 3-10
set up extended pools of, 8-4
user-defined values, 3-10
what is an extended, 13-12

B

Base Component, rule definition attribute, 11-6, 11-14, 11-19, 11-22
baseline
configuration, identify and maintain, 3-3
configuration, status, 2-7
current state of configuration, 3-9
identify current configuration, 3-11
methods to identify changes against, 11-26
show changes since, 2-8
upgrades and configuration, 3-7
batch validation parameter, 10-29
Bill of Materials (BOM)
activation sequence of lines, 13-11
dependencies, 7-1
Model node, 10-13
Model, component tracking, specify, 6-3
Option Class, component tracking, specify, 6-3
reference material, 7-1
setup, 7-1
standard item, component tracking, specify, 6-3
billing information, change, 2-10
BOM. *See* Bill of Materials.
booked, status, 2-11, 14-10
button
add to UI, 10-17
enable or disable line-level, 12-13
lookup type for, 12-13

C

catalog, Product Inventory, 14-4
change
billing information, 2-10
column label, 12-11
column sequence, 12-7
compared to baseline, types of, 11-11
default expiration date, 14-3
default location ID, 11-6
default number of rows in search results, 12-15
default values, 14-2
detecting type of, 11-9
display state of controls, 11-30
existing configuration, 2-6
existing service, 14-5
identify configuration, 11-25
installed item in the field, 2-6
instance name, 11-18
location type code, 11-6
make to configuration, 3-3
multiple instantiable child Models, A-2
name of runtime component instance, 11-18
price of item, 2-9
quantity of component, 12-3
quantity of model product, 12-4
quote, 2-9
required revalidation to substantial, A-2
seeded values, 12-7
sequence of columns, 12-12
shipping information, 2-10
since baseline configuration, 2-8
status, 2-11
status manually, 2-11
user attribute in SFM, A-2
validation of, 2-8
workflow access level, 13-4
change type
array of, 11-16
baseline, 11-15
baseline examples of, 11-13
Line Type, relationship between, 11-12
OC, 11-11, 11-13
set of Configurator, 11-13
variables mirroring baseline, 11-15
changes to installed configuration, option, 2-8
charge names, create one-time, 9-3
charges, seeded and recommended label, 12-11
checklist setup, 4-4
child Model
about deleting trackable, 15-2
import process creates structure for, 10-6

trackable, structure of, 10-7
tracking multiple instantiable, A-2
valid configuration rule combinations, 10-14
code example
configuration attributes, B-12
default location, set, B-3
Functional Companion, B-1
instance name, change, B-22
Line Type, set, B-5
code, enable action or option, 12-14
column
attribute mapping, 12-8
change sequence of, 12-7, 12-12
determine display of Line Type, 10-20
display, 12-8
hide, 12-8
label, edit, 12-11
Line Type optional seeded, 12-9
mandatory, 12-8
name, suggested renaming of, 12-7
one-time charges recommended label, 12-11
optional seeded, 12-9
provisioning action recommended label, 12-11
recommended changes to, 12-11
recurring adjustment recommended label, 12-11
recurring price list recommended label, 12-11
seeded labels of, 12-11
seeded, hide, 12-10
sequence, edit, 12-12
component
change quantity of, 12-3
unchanged, remove, 14-8
component instance
about, 11-25
in Install Base, track, 11-25
multiple, 11-25
component item
specify, 6-3
tracking, 6-3
concurrent program
Disable/Enable Refresh of a Configuration Model, 10-6
Import Configuration Models, 10-16
Populate Configuration Models, 10-6, 10-8
Refresh a Configuration Model, 10-6, 10-7
Refresh all Imported Configuration Models, 10-6, 10-7
under Populate and Refresh Configuration Models, 10-5
XDP Resubmit Failed Message, 15-5
configurable item
network link, 6-5
setup, 6-3
configuration
attribute, CZ_CONFIG_ATTRIBUTES table, 10-21
attribute, CZ_CONFIG_EXT_ATTRIBUTES table, 10-21
attributes with Install Base, 11-21
automatic behavior of, 11-31
baseline, current state of, 3-9
change existing, 2-6
changes to, identify, 11-25
complete product, 14-5
CSI_T_TRANSACTION_LINES, update, 10-25
CSI_T_TXN_LINE_DETAILS, update, 10-25
delta computation, 3-9
identify current baseline, 3-11
invalid rule combinations, 10-14
make changes to, 3-3
Model, create, 10-16
multiple component instances, 11-25
network, events in, 11-28
remove unchanged components from, 14-8
rules, define, 10-16
See also reconfigure.
show changes since baseline, 2-8
store Container Model, 3-11
update, 11-25
using attributes with IB, B-12
valid rule combinations, 10-14
Configuration Model Type, 10-7
Configurator
configuration Model, create, 10-16
Container Model settings and structure, 10-4
Container Model structure, 10-7
customize, 10-1
CZ schema customizations, 10-19
dependencies, 10-2
Functional Companions setup, 11-1

functionality, 3-7
import Container Models into, 10-5
key functionality, 3-2
launch, 3-5
map Line Type, 5-6
reference material, 10-1
setup, 10-1
validate changed configuration, 2-8
configure, new items, 2-5
connected-to option, enable, 12-14
connected-to relationships, 3-12
connection rule, adding, deleting, or changing, A-2
Connector
define rules for, 10-11
define rules using, 10-13
in Container Model, 10-11
node, delete, 15-2, A-2
reasons not to create, 10-11
requirements to create, 10-11
contact name, specify for quote, 14-3
container BOM Model
import, 10-6
See also Container Model.
tracking for components, specify, 6-3
Container Model
about rules when revalidating, 15-2
add items from Install Base to, 14-7
configuration rule requirements, 10-13
Connector requirements, 10-11
create, 6-6
define, 7-2
define as PTO BOM Model requirement, 10-4
description, 10-4
entire line, remove, 14-9
flag as, 6-3
import, 10-16
import into Configurator, 10-5
installed configurations, revalidate, 15-2
node, 10-15
node from non-trackable child, 10-15
product included in, 14-5
PTO (pick-to-order), 7-2
publish, 10-18
refresh error, 10-7
refresh warning, 10-7
requirements not to define, 10-4
requirements, verify, 6-4
root node, 10-11
settings and structure, 10-4
specify, 6-6
specify item as, 6-6
store, 3-11
structure, 10-7
structure requirements, 10-7
tracking for component item, specify, 6-3
valid and invalid examples of, 10-9
valid structure of, verify, 10-16
contract template, for quote, 14-4
convert, quote to sales order, 2-8
cost, after reconfiguration see difference in, 14-8
create
configuration Model, 10-16
Container Model, 6-6
PL/SQL procedure, 13-6
quote, 2-3, 14-2
workflow process, 13-7
CSI_T_TRANSACTION_LINES, mapping to CZ for configuration updates, 10-25
CSI_T_TXN_LINE_DETAILS, mapping to CZ for configuration updates, 10-25
currency, for quote, 14-3
current selection, option, 2-8
custom columns, 12-8
customize, user interface, 10-17
CZ Configurator Install Base, profile option, 10-18
CZ Report All Baseline Conflicts, profile option, 10-19
CZ Suppress Baseline Errors, profile option, 10-18
CZ_CONFIG_ATTRIBUTES, Table in CZ schema, 10-21
CZ_CONFIG_EXT_ATTRIBUTES, Table in CZ schema, 10-21
CZ_CONFIG_ITEMS, mapping to the Install Base schema, 10-25
CZ_DB_SETTINGS
DISPLAY_SUMMARY_FULFILLMENT_ACTION, 10-20
CZ_DB_SETTINGS, DISPLAY_SUMMARY_FULFILLMENT_ACTION, 10-20

D

data, out of synchronization, 15-2
default
  Line Type, 12-16
  set default order Line Type as, 5-5
  value, change a quote's, 14-2
default location
  code example of setting, B-3
  Functional Companion, 11-5
  ID, 11-6
  modify, 11-7
  See also location.
Default Order Status, 12-2, 12-5
Default Quote Status, 12-2, 12-5, 12-6
define
  approved rules, 2-10
  configuration Model, 10-16
  Container Model, 7-2
  Container Model as PTO BOM Model requirement, 10-4
  Container Model as untrackable, 10-4
  Container Model do not, 10-4
  extended attributes, 8-4
  Functional Companion rule, 11-19
  items as link items, 3-12
  line transaction type, 5-4
  Line Types, 10-17, 11-10, 15-3
  order transaction type, 5-5
  profile options, 10-18, 12-3
  rules for Connectors, 10-11
  rules using Connectors, 10-13
  service items, 6-3
  work item, 13-9
delete
  Connector node, 15-2, A-2
  trackable child Model, impact of, 15-2
  trackable multiple instantiable child Model, A-2
dependency setup, 4-2
disable
  Add Service action, 12-14
  line-level action, 12-13
  Split Line action, 12-14
  table-level button, 12-13
Disable/Enable Refresh of a Configuration Model concurrent program, 10-6
discontinued
  items, access, 11-27
  Line Type, 11-7
DISPLAY_SUMMARY_FULFILLMENT_ACTION CZ_DB_SETTING for UISERVER, 10-20
DISPLAY_SUMMARY_FULFILLMENT_ACTION, CZ_DB_SETTINGS, 10-20

E

edit, *See* change *or* modify.
enable
  Connected-To option, 12-14
  line-level action, 12-13
  Reconfigure action, 12-14
  Remove Product action, 12-14
  table-level button, 12-13
Entered status, note about, 2-11
error
  manage failure notification, 15-7
  manage notification, 15-3
  message, failure to accept order, 13-11
  retry failed action, 15-4
  retry failed outgoing messages, 15-5
event
  handling interface, 11-28
  in a configuration network, 11-28
  listeners, 11-30
  ON_NAVIGATE, 11-29
  POST_ACTIVATE_ACTION, 11-30
expiration
  of quote, 2-10
  specify quote's date of, 14-3
extended attribute
  create, 8-3
  define, 8-4
  global level, 8-3
  instance level, 8-4
  inventory item level, 8-3
  item category level, 8-3
  map to SFM items, 8-6
  search by, 14-6
  set up pools of, 8-4
  what is, 13-12

F

failed action, 15-3, 15-5, 15-7
find, order, 15-4
flag, item as Container Model, 6-6
flow
  business, 2-2
  setup, 4-2
fulfillment
  action, 11-10, B-5
  based on actions, 3-10
  persistence of values, 3-10
Functional Companion
  adapt, 10-17
  code examples, B-1
  default location, 11-5
  default location, about, 11-5
  default location, effects of, 11-7
  default location, modify, 11-7
  default location, specify, 11-5
  define rule, 11-19
  IBAttribute, about, 10-18
  IBAttribute, effects of, 11-23
  IBAttribute, modify, 11-23
  IBAttribute, recommended, 11-21
  IBAttribute, setup, 11-22
  instance name, about, 10-18
  instance name, change, 11-18
  instance name, modify, 11-21
  instance name, setup, 11-18
  Line Type, about, 10-17
  Line Type, modify, 11-18
  Line Type, setup, 11-10
  Line Type, solution, 11-4
  Line Type, specify, 11-9
  Line Types, effects of, 11-14
  location, 10-18
  modify or update, 11-3
  purpose of Line Type, 11-9
  rule, define, 11-6, 11-14
  setup, 11-1

G

generate user interface, 10-17
global extended attribute level, 8-3
graphic, add to UI, 10-17

H

hosting application, specify, 10-18

I

IBAttribute
  effects of, 11-23
  Functional Companion, 10-18, 11-21
  modify, 11-23
  setup, 11-22
identification, based on actions, 3-10
identify, current baseline configuration, 3-11
import
  container BOM Model, 10-6
  Container Model, 10-16
  create child Model structure, 10-6
  data with Populate and Refresh Configuration Models, 10-5
  Populate Configuration Models, 10-5
  refresh BOM with sub-models, 10-7
Import Configuration Models concurrent program, 10-16
initialization parameter
  about, 10-27
  instance, 10-28
  multiple times, specify, 10-28
  validation_context, 10-28
Install Base
  add items to Container Model from, 14-7
  add services from, 14-6
  add to Model from, 14-7
  configuration attributes with, 11-21
  dependencies, 8-1, 9-1
  design tips, 8-7
  enable network configurations, 8-3
  functionality, 3-11
  integration with, 3-7
  item, track in, 6-3
  key functionality, 3-3
  product add to, 14-6
  reconfigure, 2-6

reference material, 8-1, 9-1
remove changed lines, 14-6
retry updates to, 15-4
schema, 10-25
search, 3-4
setup, 8-1
split line, 14-6
Install Base schema
CSI_T_TRANSACTION_LINES, 10-25
CSI_T_TXN_LINE_DETAILS, 10-25
Installed Base. *See also* Install Base.
Installed Base Tracking check box, 6-4
installed configuration, update, 10-18
instance
access to, 11-27
change runtime component, 11-18
component, 11-25
description of, 10-28
minimum and maximum, PTO (pick-to-order), 10-9
multiples of configurable components, 11-25
name, Functional Companion, 11-18
reconfigure, A-1
trackable, 10-7
Instance Class, set to Link, 6-4
instance extended attribute level, 8-4
instance name
change, 11-18
code example, change, B-22
column attribute mapping, 12-8
effects of, 11-20
Functional Companion, 10-18
modify, 11-21
optional seeded column, 12-9
purpose of, 11-18
seeded and recommended label, 12-11
set up Lines page, 12-6
setup, 11-18
suggested name for column, 12-7
Instance Name or Description, suggested renaming of column, 12-7
instance node, 11-21
instances maximum, 10-7, 10-8, 10-9
instances minimum, 10-7, 10-8, 10-9
instantiable child Model
delete trackable multiple, A-2
track multiple, A-2
integration, with Install Base, 3-7
interface event, handling of, 11-28
Inventory
category, 14-4
Container Models, 6-6
define items as link items, 3-12
reference material, 6-1
setup, 6-1
tracking for component item, specify, 6-3
inventory item extended attribute level, 8-3
item
change price of, 2-9
configure new, 2-5
See also product.
item category extended attribute level, 8-3

L

launch, Configurator, 3-5
line
remove from Install Base, 14-6
split, add from Install Base, 14-6
Line Category, 12-9
Line Discount, 12-8, 12-9
Line Number, 12-9
Line Price List, 14-4
line transaction type, set up, 5-4
Line Type
action, 3-9
additional, for reconfiguration, 12-3
change type, relationship between, 11-12
code example, B-5
column attribute mapping, 12-8
column, determine display of, 10-20
default, 12-16
default action, 5-4
define, 10-17, 11-10, 15-3
discontinued, 11-7
effects of, 11-14
fulfillment action, 11-17
Functional Companion, 10-17, 11-4, 11-9
insert record for all OC change types, 11-13
known as action, 3-3

map action and, 5-6
modify, 11-18
modify variables, 11-18
MOVE, 11-17
No Action, 15-3
OC change types that map to, 11-11
optional seeded columns, 12-9
Order Management, 5-6
pricing dependency on action or, 10-19
provisionable item includes mapped, 13-11
purpose of Functional Companion, 11-9
Reprice, 15-3
seeded and recommended label, 12-11
set default order, 5-5
set the, 11-16
set up, 11-10
set up Lines page, 12-6
suggested renaming of column, 12-7
Line Type Functional Companion, 5-6, 11-14
line, remove, 14-9
line-level action, enable or disable, 12-13
Lines page
action or Line Type, 12-6
column, display, 12-8
column, hide, 12-8
instance name, 12-6
recurring adjustment, 12-6
recurring list, 12-6
seeded column, hide, 12-10
setup, 12-6
suggested column names, 12-7
unit of measure (UOM), 12-6
Link to Customer Page, 12-9
link, specify service item as, 6-3, 6-5
location
background, 11-5
default, effects of, 11-7
default, setup, 11-6
default, solution, 11-3
default, specify, 11-5
example codes of default, 11-5
for quote, 14-2
Functional Companion, 10-18, 11-5
highest node to set the ID, 11-6
modify default, 11-7
purpose of default, 11-5
location type code, 11-6
lookup code
disable, 12-14
Quoting setup, 12-2
reconfiguration setup, 12-2
lookup type, for action menus and buttons, 12-13

M

MACD_FULFILLMENT_STRUCTURES table, 10-23
maintain, current baseline configuration, 3-11
manage, failure notifications, 15-7
mandatory attribute, 13-11
map
CZ tables to Install Base tables, 10-25
extended attributes to SFM items, 8-6
item and Line Type combinations to work item, 13-11
provisionable item to Line Type, 13-11
marketing source, about, 14-3
menu notification, add function for, 13-5
menu, lookup type for, 12-13
message
failure to send, 15-6
monitor and verify sending of, 15-6
retry failed outgoing, 15-5
minus indicator, view of model, 14-5
model data, out of synchronization, 15-2
model, container, 10-4
modify
default location, 11-7
Functional Companions, 11-3
IBAttribute, 11-23
instance name, 11-21
Line Type, 11-18
See also change.

N

name, for quote, 14-2
network
configuration, enable, 8-3
event in configuration, 11-28

ON_NAVIGATE event in configuration, 11-29
partial reconfiguration and validation, 3-7
POST_ACTIVATE_ACTION event in configuration, 11-30
storing configuration model, 3-11
supporting models of, 3-4
network configuration model. *See* Container Model.
network configuration model, storing, 3-11
network link
check if node is, 11-16
service item, 6-5
service item as, specify, 6-3
No Action Line Type, 15-3
node
about deleting a Connector, 15-2
added to session, 11-26
BOM Model, 10-13
changed configuration attributes, 11-26
changed descendant, 11-26
changed location or location type code, 11-26
check if node is network link, 11-16
Container Model, 10-15
Container Model's root, 10-11
deleted from session, 11-26
determine correct action for, 11-17
from non-trackable child, 10-15
highest to set the location ID, 11-6
in Configurator, creation of model, 10-6
in parent model, reference, 10-6
instance, 11-21
replace fulfill, 5-3
value change, 11-26
notification
add function for menu, 13-5
errors, manage, 15-3
manage failure, 15-7
retry failed action, 15-4
retry failed outgoing messages, 15-5
retry Install Base updates, 15-4
with Open status, 15-5

O

OC change type, 11-13
OM. *See* Order Management.
ON_NAVIGATE, event, 11-29
one-time
create charge names, 9-3
discount charges, 9-8
pricing, setup, 9-3
one-time charges
recommended column label, 12-11
set up Lines page, 12-6
suggested renaming of column, 12-7
onLoad(), usage, 11-31
Open status, 15-5
Oracle Advanced Pricing. *See* Advanced Pricing.
Oracle Bill of Materials. *See* Bill of Materials.
Oracle Configuration Interface Object, about, 11-25
Oracle Configurator. *See* Configurator.
Oracle Install Base. *See* Install Base.
Oracle Inventory. *See* Inventory.
Oracle Order Management. *See* Order Management.
Oracle Quoting. *See* Quoting.
Oracle Service Fulfillment Manager. *See* Service Fulfillment Manager.
order
find, 15-4
place, 2-10
place a sales, 14-10
search, 15-4
set default Line Type, 5-5
Order Capture, profile option, 12-3
Order Flow-Through area, 15-4
order fulfillment, verify, 2-8
Order Management
about changing status in, 2-11
create header and line workflows, 5-3
dependencies, 5-1
design tips, 5-6
functionality, 3-2
map Line Type, 5-6
patch, apply, 5-3
profile option, 12-3
publication available to, 10-18
reference material, 5-1
replace fulfill node, 5-3
set up line transaction type, 5-4
set up order transaction type, 5-5
set up transaction types, 5-4

setup, 5-1
workflow header process, SFM enabled, 13-3
Workflow Line Process, 13-4
Order Submitted status, 14-10
order transaction type, set up, 5-5
Order Type, for quote, 14-2
Outbound Message Queue, 15-6

P

parameter, batch validation, 10-29
part number, add product, 14-4
pending order, making changes to, A-1
pick-to-order. *See* PTO.
place
an order, about status, 2-11
order, 2-10, 14-10
order, afterward, 2-11
PL/SQL procedure, create, 13-6
plus indicator, view of model, 14-5
pool, set up extended attribute, 8-4
Populate and Refresh Configuration Models
concurrent programs, 10-5
import data with, 10-5
Populate Configuration Models concurrent program, 10-6, 10-8
POST_ACTIVATE_ACTION, event, 11-30
prevent configuration model refreshing, 10-6
price
change item, 2-9
freeze, 2-10
price list, for quote, 14-4
pricing
attributes and sourcing rules, 9-9
define modifier for item or service, 9-4
define qualifiers on modifier, 9-7
dependency on Line Type or action, 10-19
disable, 10-19
discount to one-time charges, 9-8
get custom price, 9-13
items, 2-9
set up one-time, 9-3
set up one-time charge names, 9-3
primary sales group, quote, 14-3
primary salesperson, for quote, 14-3
process, business, 2-2
product
add to Container Model, 14-7
add to quote from Install Base, 14-6
complete configuration, 14-5
existing service, reconfigure, 14-5
included in Container Model, 14-5
incomplete configuration, 14-5
name, add, 14-4
part number, 14-4
removed unchanged, 14-8
to quote, add, 14-4
unchanged, remove, 14-8
unit of measure, 14-4
product catalog source, 14-4
product description, 14-4
Product Inventory category, 14-4
product name, add product, 14-4
profile option
ASO Default Salesrep, 14-3
define, 10-18, 12-3
Order Capture, 12-3
Order Management, 12-3
Quoting setup, 12-2
reconfiguration setup, 12-2
progress, an order, 13-11
proposal
create from quote, 2-10
provisionable
item, includes mapped Line Type, 13-11
item, set service item as, 6-4
item, specify service item as, 6-3
order line, status of, 13-11
Provisionable, check box, 6-4
provisioning, 2-8
provisioning action
for each quote, 2-8
recommended column label, 12-11
Provisioning Failed
status, 13-11
status on sales order form, 15-5
PTO (pick-to-order)
BOM Model, 6-7
Container Model, 7-2
minimum and maximum instances, 10-9

publication, available to Order Management, 10-18
publish, Container Model, 10-18

Q

quantity, change model product, 12-4
quote
  address for, 14-2
  agreement, 14-3
  automatic numbering, 14-3
  change, 2-9
  change default values, 14-2
  contract template, 14-4
  convert to sales order, 2-8, 14-10
  create, 2-3, 14-2
  create new version, 2-10
  currency, 14-3
  expiration, 2-10
  expiration date, 14-3
  from Install Base, add products to, 14-6
  location for, 14-2
  name for, 14-2
  Order Type, 14-2
  price list, 14-4
  pricing items in, 2-9
  primary sales group, 14-3
  primary salesperson, 14-3
  print, 2-10
  product from Install Base, add to, 14-6
  products, add to, 14-4
  proposal from, 2-10
  remove unchanged components, 14-8
  sales channel, 14-3
  Sales Online, 14-2, 14-4, 14-5, 14-6, 14-7, 14-8, 14-9, 14-10
  source for, 14-3
  specify contact name, 14-3
  submit for approval, 2-10
Quoting
  additional Line Type setup, 12-3
  change default number of rows, 12-15
  dependencies, 12-1
  design tips, 12-16
  existing service, reconfigure, 14-5
  functionality, 3-3
  key functionality, 3-2
  launch Configurator from, 3-5
  Lines page setup, 12-6
  lookup code setup, 12-2
  profile options, 12-2
  profile options, define, 12-3
  reconfiguration setup, 12-2
  reference material, 12-1
  setup, 12-1

R

reconfigure
  action, enable, 12-14
  additional Line Type setup, 12-3
  existing items, 2-6
  existing service, 14-5
  instance, A-1
  lookup code setup, 12-2
  partial network, 3-7
  profile options, 12-2
  Quoting setup, 12-2
  remove unchanged components, 14-8
  See also change.
  See also configuration.
recurring adjustment
  on Lines page, 12-6
  recommended column label, 12-11
  suggested renaming of column, 12-7
recurring list
  set up Lines page, 12-6
  suggested renaming of column, 12-7
recurring net
  set up Lines page, 12-6
  suggested renaming of column, 12-7
recurring price list, recommended column label, 12-11
refresh
  instantiable models, 10-7
  models with references, 10-7
  prevent configuration model, 10-6
Refresh a Configuration Model concurrent program, 10-6, 10-7
Refresh all Imported Configuration Models concurrent program, 10-6, 10-7

relationship, connected-to, 3-12
remove
changed lines from Install Base, 14-6
unchanged components, 14-8
Remove Product, enable, 12-14
rename, columns on the Lines page, 12-7
Reprice Line Type, 15-3
restrictions, about, A-1
revalidate, installed configurations, 15-2
row, change default number of, 12-15
rule
containing participants, 15-2
define approved, 2-10
definition attributes and values, 11-6, 11-14, 11-19, 11-22
in Container Models, 10-13
in Functional Companion, define, 11-14
invalid configuration combinations, 10-14
valid configuration combinations, 10-14
when re-validating Container Model, 15-2

S

sales channel, for quote, 14-3
Sales Online, task, 14-2, 14-4, 14-5, 14-6, 14-7, 14-8, 14-9, 14-10
sales order
convert from quote, 2-8, 14-10
place a, 14-10
scope, provisioned items, A-1
search
by extended attribute values, 14-6
for order, 15-4
Install Base, 3-4
search results, change number of rows in, 12-15
seeded columns, optional, 12-9
seeded value, change, 12-7
service
add from Install Base, 14-6
as Container Model, flag, 6-3
existing, reconfigure, 14-5
Service Fulfillment Manager
change to user attribute, A-2
dependencies, 13-2
design tips, 13-11
fails to accept order, 13-11
functionality, 3-12
key functionality, 3-3
map extended attributes to, 8-6
map work item, 13-10
PL/SQL Procedure, create, 13-6
provisioning in, 2-8
reference material, 13-1
setup, 13-1
work item, define, 13-9
workflow header process, 13-3
workflow process, create, 13-7
service item
network link, 6-3
provisionable items, 6-3
set provisionable, 6-4
setup
about, 4-2
Bill of Materials (BOM), 7-1
checklist, 4-4
configurable items, 6-3
Configurator, 10-1, 11-1
dependencies, 4-2
flow, 4-2
Functional Companions, 11-1
Install Base, 8-1
Inventory, 6-1
Order Management, 5-1
overview, 4-2
Quoting, 12-1
sequence, 4-2
Service Fulfillment Manager, 13-1
SFM. *See* Service Fulfillment Manager.
shipping information, change, 2-10
short names, of Oracle Applications, 10-18
source
for quote, 14-3
product catalog, 14-4
split line
disable action, 12-14
perform from Install Base, 14-6
status
baseline configuration, 2-7
Booked, 2-11, 14-10
change, 2-11

change manually, 2-11
Default Order Status profile option, 12-2
Default Quote Status, 12-5
Default Quote Status profile option, 12-2
Entered, 2-11
monitor area, 15-7
Open, 15-5
Order Submitted, 14-10
Provisioning Failed, 13-11
provisioning failure on sales order form, 15-5
record, 10-4
validating fulfillment, 10-29
Status Monitor area, 15-7
substantial change, required revalidation, A-2
Summary window, about, 2-8
synchronization, data, out of, 15-2

T

table-level button, enable or disable, 12-13
Tax Details, 12-8, 12-9
Total Price, 12-8
trackable
delete trackable child Model, 15-2
instance, 10-7
make child Model non-trackable, 15-2
transaction type
about, 10-17
set up, 5-4
set up order, 5-5
transaction type ID, mapping of, 11-13
triangle indicator, 14-8
triangle, about icon, 14-5

U

unchanged component
cost difference, 14-8
remove, 3-5, 14-8
Unit Adjustment Percent
column and attribute mapping, 12-8
optional seeded column, 12-9
seeded and recommended label, 12-11
suggested renaming of column, 12-7
Unit List Price
column and attribute mapping, 12-8
optional seeded column, 12-9
seeded and recommended label, 12-11
suggested renaming of column, 12-7
unit of measurement (UOM)
add product to quote, 14-4
column and attribute mapping, 12-8
set up lines page, 12-6
Unit Selling Price
column and attribute mapping, 12-8
option seeded column, 12-9
suggested column renaming, 12-7
update
installed configurations, 10-18, 11-25
See also change *or* modify.
user interface
customize, 10-17
generate, 10-17
user tasks, 14-1

V

validation, partial network, 3-7
validation_context, description of, 10-28
variable
mirroring baseline change types, 11-15
modify set of test, 11-18
setting for change methods, 11-16
textFeature, 11-20, 11-21
verify
Container Model requirements, 6-4
order fulfillment, 2-8
valid Container Model structure, 10-16

W

work item
define, 13-9
item and Line Type combinations map to, 13-11
map to action combination, 13-10
parameter, correct, 15-7
workflow
change access level, 13-4
create header and line, 5-3
process, create, 13-7

workflow header process, create SFM Enabled OM, 13-3
workflow line process, create, 13-4

X

XDP Resubmit Failed Message concurrent program, 15-5