Oracle® Configurator

Installation Guide

Release 11*i*

Part No. B13602-03

May 2005

This document describes how to install and configure Oracle Configurator and Oracle Configurator Developer.

ORACLE®

Oracle Configurator Installation Guide, Release 11*i*

Part No. B13602-03

Primary Author: Stephen R. Damiani

Contributor: Mark Sawtelle, Harriet Shanzer, Configurator Development Group

Contents

List of Examples

List of Tables

Send Us Your Comments

Oracle Configurator Installation Guide, Release 11*i*

Part No. B13602-03

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. 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 about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: czdoc_us@oracle.com
- FAX: 781-238-9896 Attn: Oracle Configurator Documentation
- Postal service:

  Oracle Corporation
  Oracle Configurator Documentation
  10 Van de Graaff Drive
  Burlington MA 01803-5146
  USA

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

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

Preface

Welcome to the *Oracle Configurator Installation Guide*. This installation guide provides explanations and instructions for tasks required to install the CZ schema, Oracle Configurator Developer, and a runtime Oracle Configurator.

Intended Audience

If you are responsible for installing Oracle Configurator, be sure you have read and understand the information in *Oracle Applications Concepts* and *Installing Oracle Applications*. *Oracle Applications Concepts* explains the technology, architecture, and terminology used with all Oracle Applications. *Installing Oracle Applications* provides instructions for installing Oracle Applications products and the CZ schema using Oracle Rapid Install.

This manual is intended for anyone installing or supporting the installation of Oracle Configurator.

Ordinarily, the tasks presented in this book are performed by one of the following people:

- System Administrator (that is, an Oracle Applications user who is assigned to the System Administrator responsibility)

  This person is responsible for administering the Oracle Applications system, including:

  - Ensuring that hardware is correctly configured
  - Installing, configuring, and maintaining production and development software
  - Ensuring that the system is backed up daily
  - Designing and maintaining system security such as system accounts

  The System Administrator provides support for problems with the system. They may perform setup and initial maintenance of the production system or advise their client's operational staff on these tasks. The System Administrator works with the project team to optimize system performance, install packaged applications environments, and convert data.

- Database Administrator

  Installs and configures the Oracle Applications database and maintains database access controls. This person also provides consultation on performance and is responsible for monitoring growth and fragmentation of the production database and ensuring database backup and recovery.

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. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation Screen readers 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, some screen readers 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 does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Structure

This guide contains the following chapters:

- Chapter 1, "Installing Oracle Configurator" provides an overview of installing and setting up Oracle Configurator and Oracle Configurator Developer. It also describes installation information specific to Multiple Language Support (MLS).
- Chapter 2, "Upgrading to this Release" contains information about Functional Companions and legacy User Interfaces that you should consider when upgrading from a previous release of Oracle Configurator.
- Chapter 3, "Oracle Configurator Servlet Considerations" describes the tasks required to install and configure the Oracle Configurator Servlet.
- Chapter 4, "Troubleshooting Servlet Installation" provides suggestions for resolving problems that may arise when installing the Oracle Configurator Servlet.
- The "Glossary"contains definitions that may be helpful while working with Oracle Configurator.

Related Documents

For more information, see the documentation for your release of Oracle Applications, Oracle10g RDBMS documentation, Oracle Configurator documentation, and the product-specific Release Notes for applications that can host a runtime Oracle Configurator.

The following documents also contain useful information related to installing and configuring Oracle Configurator and Oracle Configurator Developer:

- *Oracle Applications Concepts*
- *Installing Oracle Applications*
- *Oracle Applications User's Guide*
- *Maintaining Oracle Applications*
- *Apache 1.3 User's Guide*

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The table below lists other conventions that are also 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 new term, a term defined in the glossary, specific keys, and labels of user interface objects. Boldface type also indicates a menu, command, or option, especially within procedures
italics	Italic type in text, tables, or code examples indicates user-supplied text. Replace these placeholders with a specific value or string.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
>	The left bracket alone represents the MS DOS prompt.
$	The dollar sign represents the DIGITAL Command Language prompt in Windows and the Bourne shell prompt in Digital UNIX.
%	The percent sign alone represents the UNIX prompt.
`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.

Product Support

The mission of the Oracle Support Services organization is to help you resolve any issues or questions that you have regarding Oracle Configurator Developer and Oracle Configurator.

To report issues that are not mission-critical, submit a Technical Assistance Request (TAR) using Metalink, Oracle's technical support Web site, at:

`http://www.oracle.com/support/metalink/`

Log into your Metalink account and navigate to the Configurator TAR template:

1. Choose the **TARs** link in the left menu.
2. Click on **Create a TAR**.
3. Fill in or choose a profile.
4. In the same form:
   a. Choose **Product**: Oracle Configurator or Oracle Configurator Developer
   b. Choose **Type of Problem**: Oracle Configurator Generic Issue template
5. Provide the information requested in the iTAR template.

You can also find product-specific documentation and other useful information using Metalink.

For a complete listing of available Oracle Support Services and phone numbers, see:

`http://www.oracle.com/support`

Troubleshooting

Oracle Configurator Developer and Oracle Configurator use the standard Oracle Applications methods of logging to analyze and debug both development and runtime issues. These methods include setting various profile options and Java system properties to enable logging and specify the desired level of detail you want to record.

For general information about the logging options available when working in Configurator Developer, see the *Oracle Configurator Developer User's Guide*.

For details about the logging methods available in Configurator Developer and a runtime Oracle Configurator, see:

- The *Oracle Applications System Administrator's Guide* for descriptions of the Oracle Applications Manager UI screens that allow System Administrators to set up logging profiles, review Java system properties, search for log messages, and so on.
- The *Oracle Applications Supportability Guide*, which includes logging guidelines for both System Administrators and developers, and related topics.
- The Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

1 Installing Oracle Configurator

This chapter presents the following topics:

1.1 Overview

Oracle Configurator consists of the following:

- The CZ schema: A subschema within the Oracle Applications database that stores configuration model data.
- Oracle Configurator Developer: An application based on the Oracle Applications (OA) Framework that is used to develop a configuration model and a configurator.
- The runtime Oracle Configurator: The end-user environment in which users configure orderable products or services.

If you are installing Oracle Applications Release 11*i* for the first time, the CZ schema, Oracle Configurator Developer, and the runtime Oracle Configurator are installed by running Oracle Rapid Install. For details, see Section 1.2, "Oracle Rapid Install" on page 1-1.

If you are upgrading an existing Oracle Configurator installation, see Chapter 2, "Upgrading to this Release".

> **Warning:** Before beginning to implement your configurator project, attend training in Oracle Configurator Developer, read the *Oracle Configurator Developer User's Guide* and review the *About Oracle Configurator* documentation for this release on Metalink, Oracle's technical support Web site.

1.2 Oracle Rapid Install

Oracle Rapid Install is an automated process that installs Oracle Applications Release 11*i*, and Oracle Internet Application Server (*i*AS). It also installs the Apache Web server and supporting software. When running Rapid Install, an installation wizard guides

you through the Oracle Applications installation process. You select the product(s) you want to install and Rapid Install automatically selects and installs any dependent products. You can install up to three database instances at the same time.

The information you supply in the Rapid Install wizard is captured in a configuration file, which you store for use during the various stages of your installation or upgrade.

Rapid Install provides default values for some profile options and Oracle Configurator Servlet properties. After running Rapid Install, be sure all profile options and system properties that have default values are set correctly for your installation, and set up any that do not have default values. For details, see Section 1.3, "Additional Setup Tasks" on page 1-2.

For more information about Oracle Rapid Install, see *Installing Oracle Applications*.

Note: If you are implementing Multiple Language Support (MLS), read Section 1.6, "Installation and Setup Considerations for Multiple Language Support" on page 1-17 before running Rapid Install.

1.3 Additional Setup Tasks

After running Oracle Rapid Install or upgrading to the latest version of Oracle Configurator, perform the following, if applicable:

1. Review all Oracle Configurator and Configurator Developer profile options and, if necessary, modify them for your installation.

   See Section 1.3.1, "Set Profile Options" on page 1-3.

2. Verify that Oracle Rapid Install has set up the server correctly (new installations only).

   See Section 3.3, "Verifying Apache and JServ Setup" on page 3-2.

3. Define users in Oracle Applications and assign them to at least one of the predefined responsibilities that provides access to Oracle Configurator Developer. This task is typically performed by the System Administrator.

   For details about defining users and general information about responsibilities, see the *Oracle Applications System Administrator's Guide*.

   The predefined Oracle Configurator Developer responsibilities are described in the *Oracle Configurator Implementation Guide*.

4. If required, create custom Oracle Applications responsibilities. Only the System Administrator can create new responsibilities.

   Creating responsibilities is described in the *Oracle Applications System Administrator's Guide*.

5. Verify that Oracle Configurator and Oracle Configurator Developer were installed successfully and are set up set up correctly.

   See Section 1.4, "Test Your Oracle Configurator and Oracle Configurator Developer Installation" on page 1-15.

6. If you are implementing Multiple Language Support (MLS), see Section 1.6, "Installation and Setup Considerations for Multiple Language Support" on page 1-17.

1.3.1 Set Profile Options

To utilize some Oracle Configurator Developer functionality or run the runtime Oracle Configurator within other Oracle Applications such as Order Management, you must set some profile options and decide whether default values for others are appropriate for your installation. The System Administrator responsibility has access to all Oracle Applications profile options.

The profile options that are used by the runtime Oracle Configurator are listed in Table 1–1 on page 1-3. Only the System Administrator can view and update these profile options.

The profile options that are used by Oracle Configurator Developer are listed in Table 1–2 on page 1-5.

For information about setting profile options, see the *Oracle Applications User's Guide*.

> **Note:** In the tables below, if the Default Value column is empty, the corresponding profile option does not have a default value.

Table 1–1 Runtime Oracle Configurator Profile Options

Profile Option	User	System Administrator				Requirements	Default Value
	User	User	Resp	App	Site	Required?	
BOM: Configurator URL of UI Manager on page 1-6	-	X	X	X	X	Required with Order Management, *i*Store, TeleSales, and SalesOnLine	
CZ: Auto-Expire Discontinued IB Trackable Items on page 1-6	-	X	X	X	X	Optional	Yes
CZ: Configurator Install Base on page 1-7	-	X	X	X	X	Optional	oracle.apps.cz.dio.config.OracleInstalledBase
CZ: Create Item Type Name Method on page 1-8	-	0	0	0	X	Required with Oracle Bills of Material	Item Catalog Description
CZ: BOM Tree Expansion State on page 1-7	-	X	0	0	X	Optional	One level
CZ: Fail BV if Configuration Changed on page 1-8	-	X	X	X	X	Required with Order Management	No
CZ: Fail BV if Input Quantities Not Maintained on page 1-8	-	0	X	X	X	Required with Order Management	Yes
CZ: Generic Configurator UI Max Child Rows on page 1-9	-	0	0	0	X	Required	50

Table 1–1 (Cont.) Runtime Oracle Configurator Profile Options

Profile Option	User	System Administrator				Requirements	Default Value
	User	User	Resp	App	Site	Required?	
CZ: Generic Configurator UI Type on page 1-9	-	X	0	0	0	Optional	Java Applet
CZ: Include Unchanged Install Base Items on page 1-9	-	0	0	0	X	Optional	Yes
CZ: Only Create CZ Config Items for Selected Nodes on page 1-10	-	0	0	0	X	Optional	No
CZ: Populate Decimal Quantity Flags on page 1-11	-	0	0	0	X	Optional	No
CZ: Publication Lookup Mode on page 1-12	-	X	X	X	X	Optional	Production
CZ: Publication Usage on page 1-12	-	X	X	X	X	Optional	Any Usage
CZ: Report All Baseline Conflicts on page 1-12	-	X	X	X	X	Required with Oracle Install Base	No
CZ: Skip Validation Procedure on page 1-13	-	0	0	0	X	Optional	
CZ: Suppress Baseline Errors on page 1-13	-	X	X	X	X	Required with Oracle Install Base	No
CZ: Use Alternate Retraction Algorithm Before Structure Changes on page 1-14	-	0	0	0	X	Optional	No
CZ: Use Generic Configurator UI on page 1-15	-	-	X	-	X	Required with *i*Store or Quoting	Yes
GMA: Default Language on page 1-15	-	X	X	X	X	Required	US
ICX: Language on page 1-15	-	X	0	0	X	Required	Required
Key:	X	The profile option can be set at this level by the System Administrator.					
	0	The profile option cannot be set at this level.					
	-	A Configurator Developer user cannot view or change the value of this profile option.					

Table 1–2 on page 1-5 lists the profile options that are used by Oracle Configurator Developer. The System Administrator can specify a value for these profile options at

various levels. A Configurator Developer user can modify the default User-level value for each option by changing various settings in Oracle Configurator Developer. In other words, changing one of these settings does not affect any other Configurator Developer users.

For example, to change the User-level setting for CZ: BOM Node Display Name, a Configurator Developer user modifies the BOM Node Display Names setting. This setting appears in the General area of the Workbench. Other settings that a Configurator Developer user can modify appear in the Preferences page.

For more information about updating settings in Configurator Developer, see the *Oracle Configurator Developer User's Guide*.

Table 1–2 Oracle Configurator Developer Profile Options

Profile Option	User	System Administrator				Requirements	Default Value
	User	User	Resp	App	Site	Required?	
CZ: BOM Node Display Name on page 1-7	+	X	X	X	X	Required	Description
CZ: BOM Structure Display Method on page 1-7	+	X	X	X	X	Required	Description
CZ: Custom Initialization Parameters on page 1-8	+	X	X	X	X	Optional	
CZ: Effectivity Date Filter on page 1-8	+	X	X	X	X	Optional	All
CZ: Enable Creation of Functional Companions on page 1-8	0	0	0	0	X	Optional	No
CZ: Non-BOM Node Display Name on page 1-10	+	X	X	X	X	Required	Name
CZ: Non-BOM Structure Display Method on page 1-10	+	X	X	X	X	Required	Name
CZ: Number of Table Rows Displayed on page 1-10	+	X	X	X	X	Required	25
CZ: Number of Rows Displayed in Hierarchical Tables on page 1-10	-	0	0	0	X	Optional	50

Table 1–2 (Cont.) Oracle Configurator Developer Profile Options

Profile Option	User	System Administrator				Requirements	Default Value
	User	User	Resp	App	Site	Required?	
CZ: Require Locking on page 1-13	-	0	0	0	X	Required	Yes
Help System Root on page 1-15	-	X	X	X	X	Required to display online help	CZ:CONTENTS (at Application level)
Key:	X	The profile option can be set at this level by the System Administrator.					
	0	The profile option cannot be set at this level.					
	+	A Configurator Developer user can view and change the value of this profile option.					
	-	A Configurator Developer user cannot view or change the value of this profile option.					

1.3.1.1 BOM: Configurator URL of UI Manager

This profile option indicates the Oracle Configurator Servlet URL, which is where the Oracle Configurator Servlet resides. This profile option must be set correctly for the host application to locate the Oracle Configurator Servlet.

The person installing Oracle Applications supplies this URL when running Oracle Rapid Install. Oracle Rapid Install uses this information as the default value for BOM: Configurator URL of UI Manager.

If you recently upgraded to a new version of Oracle Configurator, verify that this profile option is set correctly.

If you want to run Oracle Configurator on a different machine than the one on which Oracle Applications runs, define the servlet property `cz.runtime.use_dedicated_jvm`. See Section 3.4.1, "Descriptions of Oracle Configurator Servlet Properties" on page 3-7.

> **Note:** Setting this profile option is not required if you are installing a runtime Oracle Configurator running in a custom Web application. In this case, the person setting up the application that will be hosting the runtime Oracle Configurator must specify the URL of the Oracle Configurator Servlet, and then post the initialization message to that URL.

All URLs in your profile options should be specified with the URL format: *machine_name.domain*:*port_number*; where *machine_name* is the name of the server machine, *domain* is your domain name, and *port_number* is the port where your service is running. The Apache server port is typically 880*n*. For example:

```
http://appsmachine.appsdomain:8800/configurator/oracle.apps.cz.servlet.UiServlet
```

1.3.1.2 CZ: Auto-Expire Discontinued IB Trackable Items

This profile option controls whether an item's status automatically changes to Expired in Oracle Install Base when an Oracle Configurator end user is reconfiguring an installed instance and deselects the item.

The default value is `Yes`, which means items are automatically set to Expired in Oracle Install Base when they are removed from a configuration. This profile option can be set at the Site, Application, Responsibility, and User levels.

For more information about the integration between Oracle Install Base and Oracle Configurator, see the *Oracle Telecommunications Service Ordering Process Guide*. For more information about Oracle Install Base, see *Oracle Install Base Concepts and Procedures*.

1.3.1.3 CZ: Automatically Validate on Exit

This profile option controls validation behavior of the runtime Oracle Configurator when an end user ends a configuration session. The default value is `Always Validate on Exit`.

1.3.1.4 CZ: BOM Node Display Name

This profile option stores the value of the BOM Node Display Names setting. This setting appears in the General area of the Workbench in Oracle Configurator Developer. The default value is Description.

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

1.3.1.5 CZ: BOM Structure Display Method

This profile option stores the value of the BOM Structure Nodes setting. This setting appears in the Oracle Configurator Developer Preferences page.

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

1.3.1.6 CZ: BOM Tree Expansion State

This profile option controls the initial expansion level of the Model tree when the Generic Configurator UI that appears as a hierarchical table is launched from a host application.

If this profile option is not defined or is set to One Level (this is the default), only the root node and its children are displayed. In this case, all other branches are initially collapsed. Any node that contains children is a branch; therefore Model References, Components, and BOM Option Classes are usually branches that can be expanded and collapsed.

Set this profile option to Full to expand all branches of the Model tree when the configuration session begins.

You can set this profile option at the Site level only. For more information about Generic Configurator UIs, see the *Oracle Configurator Implementation Guide*.

1.3.1.7 CZ: Configurator Install Base

This profile option enables Oracle Configurator to interact with an installed base repository by specifying the Java class to call when an end user configures trackable components.

The default value is `oracle.apps.cz.dio.config.OracleInstalledBase` which allows Oracle Configurator to integrate with Oracle Install Base. Oracle Install Base is the only repository currently supported for integration with Oracle Configurator.

This option can be set at the Site, Application, Responsibility, and User levels.

For more information about trackable components and integration with Oracle Install Base, see the *Oracle Telecommunications Service Ordering Process Guide*.

1.3.1.8 CZ: Create Item Type Name Method

When you import a BOM Model, Item Catalog Groups appear as Item Types in Configurator Developer (that is, in the CZ schema's Item Master). This profile option determines the default Item Type Name for each imported Item Catalog Group. You can create Item Type names using either Item Catalog Group descriptions (defined in Oracle Inventory) or Item Catalog Group concatenated flexfield segments. The default is Item Catalog Description.

This profile option can be set at the Site level only.

1.3.1.9 CZ: Custom Initialization Parameters

This profile option stores the value of the Custom Initialization Parameters setting in the Oracle Configurator Developer Preferences page.

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

1.3.1.10 CZ: Effectivity Date Filter

This profile option stores the value of the Effectivity Date Filter setting in the Oracle Configurator Developer Preferences page.

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

1.3.1.11 CZ: Enable Creation of Functional Companions

Set this profile option to `Yes` if you want to be able to create new Functional Companions and edit them in Configurator Developer. This profile option is set to `No` by default.

Enable this feature only if you need to create or modify Functional Companions in a DHTML User Interface that was created in a previous release of Configurator Developer. If you are using a User Interface generated with the HTML-based version of Oracle Configurator Developer, use Configurator Extensions, not Functional Companions. See Section 2.4, "Maintaining or Migrating Functional Companions" on page 2-2.

See the *Oracle Configurator Developer User's Guide* for more information about Configurator Extensions.

This profile option can be set only at the Site level.

1.3.1.12 CZ: Fail BV if Configuration Changed

Use this profile option to determine whether the Batch Validation process fails when validating a saved configuration that has changed. The default value is `No`. Batch Validation occurs when booking an order in Oracle Order Management.

Set this option to `Yes` if you want Batch Validate to notify the end user when a previously saved quote or order has changed. In this case, the message that appears contains the item name, description, current quantity, and new quantity.

This profile option can be set at the Site, Application, Responsibility, and User Levels.

1.3.1.13 CZ: Fail BV if Input Quantities Not Maintained

This profile option determines whether the Batch Validation (BV) process fails if any input quantities change during the BV session. For example, Oracle Order Management passes a quantity of 2 for item A and 3 for item B. Item A is successfully selected first in the BV session and has a quantity of 2. Item B's quantity of 3 is then asserted. A Numeric rule propagates and subtracts 1 from the quantity of A. Item A

now has a quantity of 1, which does not match the input value. If this profile option is set to `Yes`, Batch Validation fails and return a validation status of false.

The default value is `Yes`, which means quantities are checked against the input quantities during Batch Validation. This profile option can be set at the Site level only.

1.3.1.14 CZ: Generic Configurator UI Max Child Rows

This profile option controls the maximum number of rows that are displayed at a time in the Generic Configurator User Interface. For example, this profile option is set to 50 (this is the default value). At runtime, an end user expands a BOM Option Class that has 200 Standard Items. The Generic Configurator UI displays 50 of the Items, and provides navigation controls that enable the user to display the next 50 Items.

Customizing the value of this profile option may improve performance of the Generic Configurator UI when the Model has many child Items.

This profile option can be set at the Site level only.

For details about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

1.3.1.15 CZ: Generic Configurator UI Type

This profile option specifies the type of Generic Configurator User Interface to display when configuring a BOM Model item for which no matching publication exists. If your host application is Forms-based (such as Oracle Order Management), use this profile option to control whether to use the Java Applet UI or the HTML Hierarchical Table UI. HTML-based host applications such as *i*Store use the HTML Hierarchical Table UI, regardless of how this profile option is set.

Valid values for this profile option are `Java Applet` and `HTML Hierarchical Table`. The default is `Java Applet`.

For details about the available Generic Configurator UI types and when they are used, see the *Oracle Configurator Implementation Guide*.

This profile option can be set at the User level only.

1.3.1.16 CZ: Hide Focus in Generic Configurator UI

This profile option controls whether the Focus column appears in the Generic Configurator User Interface. The Focus column displays an icon that enables an end user to view only a specific part of the Model structure. By default, this profile option is set to No, which means the Focus column does appear.

This profile option can be set at the Site level only.

For details about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

1.3.1.17 CZ: Include Unchanged Install Base Items

When an Oracle Install Base user makes a request to reconfigure an installed instance, this profile option controls what items are returned as lines on the Quote (in Oracle Quoting) or Sales Order (in Oracle Order Management).

Accept the default value of `Yes` if you want all items to appear in the Quote or Sales Order, regardless of whether they have changed in Install Base. Set this profile option to `No` if you want only items that have changed to appear in the Quote or Sales Order. Setting this profile option to `No` may improve runtime performance because a smaller set of items is returned to the host application.

This profile option can be set at the Site level only.

> **Note:** When the value of this profile option changes, you must restart the Oracle Configurator Servlet for configurations to use the new setting.

For more information about the integration between Oracle Install Base and Oracle Configurator, see the *Oracle Telecommunications Service Ordering Process Guide*.

1.3.1.18 CZ: Non-BOM Node Display Name

This profile option stores the value of the Non-BOM Node Display Names setting, which appears in the General area of the Workbench. The default is Name.

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

1.3.1.19 CZ: Non-BOM Structure Display Method

This profile option stores the value of the Non-BOM Structure Nodes setting, which appears in the Oracle Configurator Developer Preferences page.

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

1.3.1.20 CZ: Number of Table Rows Displayed

This profile option stores the value specified for the Number of Table Rows Displayed setting, which appears in the Oracle Configurator Developer Preferences page. The default is 25.

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

1.3.1.21 CZ: Number of Rows Displayed in Hierarchical Tables

This profile option controls the maximum number of rows that are displayed by default in Configurator Developer pages that display data in a hierarchy. These pages include all areas of the Repository and the Workbench. The default value is 50.

If a page contains more rows than the number specified here, Configurator Developer provides links at the top and bottom of the page so users can view the additional rows.

1.3.1.22 CZ: Only Create CZ Config Items for Selected Nodes

This profile option controls whether the runtime Oracle Configurator creates CZ Config Items for options that are included in a saved configuration.

The default value of this profile option is `No`, which means CZ Config Items are created for all:

- BOM Models, regardless of whether they are selected
- Integer Features, Decimal Features and Text Features, regardless of whether they have any user input value
- Instantiable Components, even if they have no descendants that are selected or have user inputs
- Totals and Resources

If the value of this profile option is `Yes`, CZ Config Items are created for:

- BOM Models, but only when any of the following are true:

  - They are selected
  - They have descendants that are selected
  - They are the target of a Connector
- Integer Features, Decimal Features and Text Features, only if they have user input values
- Instantiable Components, only if they have descendants that are selected or have user inputs, or are the target of a Connector

CZ Config Items are never created for Totals or Resources when this profile option is set to `Yes`.

Setting this option to `Yes` may improve performance when saving large configuration models, or a configuration that has many initial BOM Model instances.

This profile option can be set at the Site level only.

> **Note:** When the value of this profile option changes, you must restart the Oracle Configurator Servlet for configurations to use the new setting.

1.3.1.23 CZ: Populate Decimal Quantity Flags

In Oracle Inventory, an Item can be defined as accepting a decimal quantity. This profile option controls whether BOM Standard Items that accept decimal quantities are imported into the CZ schema as allowing end users to enter a decimal quantity in a generated User Interface. For details about how the Generic Configurator User Interface uses this profile option, see the *Oracle Configurator Implementation Guide*.

This option can be set at the Site level only.

Table 1–3 describes the effect of setting this profile option.

Table 1–3 CZ: Populate Decimal Quantity Flags Profile Option

Value	Effect
No	This is the default value. All items are imported as allowing only integer input, regardless of how they were defined in Oracle Inventory.
Yes	All items are imported as allowing either decimal or integer input, depending on how they were defined in Oracle Inventory.

If your host application does not support input of decimal quantities, it is recommended that you set the value of this profile option to No and import all BOM Standard Items as allowing only integer values.

If your sales order system does support input of decimal quantities, set the value of this profile option to Yes and then import new BOM Models or refresh and republish existing models to use the new setting.

When the value of this profile option changes, you must perform the following for existing publications to use the new setting:

- Refresh all imported BOM Models
- Republish existing Model publications

- Restart the Oracle Configurator Servlet

  See Chapter 3, "Oracle Configurator Servlet Considerations".

This profile option affects the behavior of the Import and Import Refresh concurrent programs. The internal name of this profile option is CZ_IMP_DECIMAL_QTY_FLAG. See the *Oracle Configurator Implementation Guide* for details.

1.3.1.24 CZ: Publication Lookup Mode

When publishing a configuration model, an Oracle Configurator Developer user specifies a publication mode to control its availability to hosting applications. Oracle Applications products such as Order Management and *i*Store use this profile option to determine the publication mode to use when selecting a publication.

You can set this profile option to either `Production` or `Test` at the Site level only. The default value is `Production`.

> **Note:** The value of this profile option is cached at runtime. Therefore, when the profile option's value changes, log out of Oracle Applications before starting another configuration session to be sure the new value is used.

1.3.1.25 CZ: Publication Usage

When publishing a configuration model, an Oracle Configurator Developer user specifies one or more Usages to control the publication's availability to hosting applications. Oracle Applications products such as Order Management and *i*Store use this profile option to determine the Usage name to use when selecting a publication.

Valid values for this profile option include any Usage names defined in Oracle Configurator Developer. The default value of this profile option is `Any Usage`, which does not limit the availability of publications based on Usages.

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

1.3.1.26 CZ: Report All Baseline Conflicts

If your Oracle Configurator end users can reconfigure installed instances, this profile option determines what Oracle Configurator considers to be a conflict between the instance the end user wants to reconfigure and the baseline configuration that exists in Oracle Install Base. (In other words, when the differences between the two are significant.)

If this option is set to `Yes`, Oracle Configurator displays a message when the baseline has changed, even if the changes are minor and compatible with the instance that the end user wants to reconfigure. The message that appears lists the differences between the baseline and the selected instance's configuration.

If this option is set to `No`, Oracle Configurator displays a message only when the baseline has changed *and* the changes are not compatible with the instance that the end user wants to reconfigure. This is the default value.

You can set this profile option at the Site, Application, Responsibility, and User levels.

See also Section 1.3.1.29, "CZ: Suppress Baseline Errors" on page 1-13.

For more information about the integration between Oracle Install Base and Oracle Configurator, see the *Oracle Telecommunications Service Ordering Process Guide*.

1.3.1.27 CZ: Require Locking

This profile option controls whether Models and UI Content Templates must be locked before you can edit them or perform certain global operations in Configurator Developer. Global operations include publishing and refreshing a Model, and generating logic. The default value is `Yes`. It is recommended that you require locking in Configurator Developer to ensure the integrity of your configuration model data.

This profile option can be set at the Site level only.

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

1.3.1.28 CZ: Skip Validation Procedure

This profile option allows the Oracle Order Management batch validation process to complete more quickly by allowing parts of the process to be skipped. You may want to do this if, for example, your configuration model is large or has many complex rules, and batch validation takes a considerable amount of time to run.

To use this profile option, you must define a PL/SQL callback function that checks specific criteria of your configuration. When this function returns a value of `True`, the Batch Validation process does not perform all of its typical tasks, such as restoring the configuration and validating any input values.

Enable this profile option only if one of the following conditions applies:

- None of your configuration models include Configurator Extensions (formerly called Functional Companions) that cause the validity of a configuration to depend on data external to the published Configurator model. In this case the value of the profile option should be the name of a PL/SQL function that always returns `true`.
- External data dependencies such as the one described above exist, but you are able to detect programmatically whether that external data has changed since a given configuration was last validated. In this case, the value of the profile option should be the name of a PL/SQL function that returns `true` only if the external data is known to be unchanged.

If neither of these conditions apply, you should not provide a value for this profile option. In this case, the batch validation process always performs a complete validation.

To enable this profile option, specify the name of your PL/SQL callback function, using the following format:

```
package_name.procedure_name
```

This profile option can be set at the Site level only.

For more information about skipping batch validation, and the PL/SQL callback function you must define to do this, see the *Oracle Configurator Implementation Guide*.

1.3.1.29 CZ: Suppress Baseline Errors

If your Oracle Configurator end users can reconfigure installed instances, this profile option determines whether Oracle Configurator displays a message when the baseline configuration of the instance that exists in Oracle Install Base has changed, even though the changes *are* compatible with the configuration that the end user wants to reconfigure. (In other words, the baseline's Instance Revision Number has changed in Oracle Install Base.)

When an end user reconfigures an instance of a configuration, Oracle Configurator compares it to the baseline configuration in Oracle Install Base. The baseline is specified by the Instance Revision Number of the component's installed instance. If the Instance Revision Number of the component being reconfigured does not match the installed component's Instance Revision Number, validation errors are generated, because the installed and the new instances do not have the same baseline. This can happen if you are restoring a saved configuration after another configuration based on the same baseline has been accepted into the Install Base data repository.

If this option is set to `Yes`, Oracle Configurator does *not* display a message when the installed instance's Instance Revision Number has changed.

The default value is `No`, which means Oracle Configurator displays a message when the component's Instance Revision Number has changed. This message lists the differences between the baseline configuration and the instance the end user wants to reconfigure.

If the baseline configuration and the instance that the end user wants to reconfigure are *not* compatible (that is, the changes are significant), Oracle Configurator displays a message regardless of how you set this profile option.

You can set this profile option at the Site, Application, Responsibility, and User levels.

> **Note:** Regardless of how you set this profile option or CZ: Report All Baseline Conflicts, the configuration always contains the latest baseline information when the Oracle Configurator session begins.

See also Section 1.3.1.26, "CZ: Report All Baseline Conflicts" on page 1-12.

For more information about the integration between Oracle Install Base and Oracle Configurator, see the *Oracle Telecommunications Service Ordering Process Guide*.

1.3.1.30 CZ: Use Alternate Retraction Algorithm Before Structure Changes

This profile option controls whether Oracle Configurator uses an internal retraction method when an end user changes the configuration by:

- Adding or deleting a component
- Creating or clearing a connection

If this profile option is set to `Yes`, Oracle Configurator retracts all end user selections at the same time before updating the configuration. Enabling this profile option may improve performance in large Models when a user performs one of the operations listed above. By default, this profile option is set to `No` and Oracle Configurator retracts each selection separately before updating the configuration. For more information, see the section on user requests in the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

This profile option can be set at the Site level only.

> **Note:** When the value of this profile option changes, you must restart the Oracle Configurator Servlet for configurations to use the new setting.

1.3.1.31 CZ: Use Generic Configurator UI

This profile option is used only by Oracle *i*Store and Oracle Quoting. It controls whether these applications consider the Generic Configurator User Interface a valid user interface.

This profile option is used by some Oracle Configurator APIs, like cz_cf_api.ui_for_item. Oracle *i*Store and Oracle Quoting call this API to decide whether or not an item is configurable and, if it is, which UI to display when an end user makes a request to configure it.

The default value of this profile option is Yes. In this case, if there is a published UI, cz_cf_api.ui_for_item returns the published UI. If the profile option is set to Yes but no published UI exists, cz_cf_api.ui_for_item returns the Generic Configurator UI. If the item is not a Model, the API returns null, which tells the calling application that the item is not configurable.

If this profile option is set to No and a published UI is found for the item, cz_cf_api.ui_for_item returns the published UI. If this profile option is set to No and there is no published UI, cz_cf_api.ui_for_item returns null. In this case, the item will not be configurable from the host application (for example, the Configure button or action will not be available for the item).

The default value of this profile option is `Yes` and it can be set at the Site and Responsibility levels.

For information about the available types of Generic Configurator UIs, see Section 1.3.1.15, "CZ: Generic Configurator UI Type" on page 1-9.

For details about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

1.3.1.32 GMA: Default Language

The default value for this profile option is set at the Site level by Oracle Rapid Install and is the base language of your Oracle Applications instance. This profile option is relevant if you are using Multiple Language Support (MLS). See Section 1.6.1, "Configuring Oracle Configurator Developer for Multiple Language Support" on page 1-17.

1.3.1.33 Help System Root

This profile option determines which product's context sensitive help content is displayed when a user clicks the global Help link in Oracle Applications. When you install Oracle Configurator (which includes Oracle Configurator Developer), this profile option is set to "CZ:CONTENTS" at the Application level by default.

1.3.1.34 ICX: Language

This user profile option is set at the Site level by Oracle Rapid Install. Its value is the base language of your Oracle Applications instance, which is stored in FND_LANGUAGES.LANGUAGE_CODE.

1.4 Test Your Oracle Configurator and Oracle Configurator Developer Installation

This section describes how to access Oracle Configurator Developer, verify that you can perform basic functions, and launch the runtime Oracle Configurator to test your installation and connectivity.

Before performing the steps below, review Section 1.5, "Web Browser Requirements" on page 1-16.

For details about building Model structure, defining rules, generating a User Interface, and unit testing, see the *Oracle Configurator Developer User's Guide*.

To test your Oracle Configurator and Oracle Configurator Developer installation:

1. Log in to Oracle Applications.
2. From the Oracle E-Business Suite Home page, select one of the predefined Oracle Configurator Developer responsibilities, and then select **Oracle Configurator Developer** from the list of available applications.
3. If your installation was successful, the Main area of the Repository appears.

   If you receive an error, set the profile option FND: Diagnostics to `Yes`. When you launch Configurator Developer again, the error message will contain more detailed information to help you resolve the issue.
4. Create a Model or open an existing Model for editing.

   For details about this step and the remaining steps in this section, see the *Oracle Configurator Developer User's Guide*.
5. Build Model structure. For example, create Components, Features, Feature Options, and so on.
6. Navigate to the Rules area of the Workbench, and then define one or more rules.
7. Generate logic for the Model. You do this from the General area of the Workbench.
8. Navigate to the User Interface area of the Workbench, and then generate a User Interface.
9. Verify that you can unit test the configuration model by launching the User Interface. To do this:
   a. From the Structure, Rules, or User Interface area of the Workbench, click **Test Model**.
   b. Select **User Interface**, and then select the User Interface you want to test from the list.
   c. Click **Finish**. If your system is set up correctly, the User Interface you selected appears in a runtime Oracle Configurator window.

      If you receive an error, contact Oracle Support Services. For details, see "Product Support" on page xiii.

1.5 Web Browser Requirements

Any Web browser running a generated Oracle Configurator User Interface must be set to display and use JavaScript and Cascading Stylesheets, and must be able to accept cookies. These requirements are met by Netscape 4.06 or later and Internet Explorer 4.0 with Service Pack 2 or later.

For more information, see the Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

1.6 Installation and Setup Considerations for Multiple Language Support

Multiple Language Support (MLS) enables you to create a Model and one or more user interfaces in your base language and then display the runtime UI in any language in which you do business.

Before running Oracle Rapid Install, review *Oracle Applications Concepts* for background information on language support in Oracle Applications, including how to select languages, character sets, and territory values.

Refer to *Installing Oracle Applications* for information about Rapid Install and National Language Support (NLS) in Oracle Applications.

1.6.1 Configuring Oracle Configurator Developer for Multiple Language Support

If you are implementing Multiple Language Support (MLS), you may want to change the language in which all prompts, menus, instructional text, and so on appear in Configurator Developer. To access the setting that enables you to do this, log into Oracle Applications, and then click the global Preferences link.

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

1.6.2 Setting Up the Runtime Oracle Configurator for MLS

To set up the runtime Oracle Configurator to support MLS, set the profile option ICX: CLIENT_IANA_ENCODING to "UTF-8". This is the default method used to display the Oracle Configurator session character set. This enables you to display more than one language in the same UI page at runtime.

1.6.2.1 Configuring Browsers for MLS

The following procedures configure your browser for using fonts compatible with MLS.

Internet Explorer

If your browser is Microsoft Internet Explorer 5.0 or higher, begin by visiting a Web site that uses the fonts appropriate to the language you want to use. Before you reach this site, a message box will appear, asking you whether you want to download a font driver for the language. Click Yes to download and install the font driver automatically. When the download is complete, close and then restart Internet Explorer. Then choose View > Encoding > More, and select the character set you want to use for the language that you specified.

Netscape Navigator

If your browser is Netscape Navigator 4.6 or higher, and is an English language version with foreign language fonts installed, use the following procedure:

1. Choose **Edit > Preferences > Navigator > Languages**, and the languages that you want to use at runtime.
2. Choose **View > Character Set** and the specific character set you want to use.
3. Choose **View > Character Set > Set Default Character Set**.
4. Choose **Edit > Preferences > Appearance > Fonts** and select the desired code set from the For the Encoding list (for example, Japanese).

5. In the option group for respecting document-specified fonts, choose the option that uses your default font setting, and ignores the document-specified fonts.

1.7 Required Patches

After installing Oracle Applications, you must run `adpatch` to apply the latest patches to your Oracle Applications Release 11*i* environment. For the latest required patches, contact Oracle Support or go to Metalink, Oracle's technical support Web site.

See *Maintaining Oracle Applications* for information about applying patches.

2

Upgrading to this Release

This chapter contains information you should consider when upgrading from a previous release of Oracle Configurator.

This chapter contains the following sections:

2.1 Introduction

If you are upgrading from a previous version of Oracle Configurator Release 11*i*, run `adpatch` to apply the latest patches to your Oracle Applications Release 11*i* environment. For the latest required patches, go to Metalink, Oracle's technical support Web site. See *Maintaining Oracle Applications* for information about applying these patches.

For additional information about this release of Oracle Configurator and Oracle Configurator Developer, see the *About Oracle Configurator* documentation on Metalink, Oracle's technical support Web site.

> **Note:** Upgrading to a new version of Oracle Configurator does not prevent end users from restoring configurations that were saved using a previous version of the software. Additionally, restoring such a saved configuration does not adversely affect the configuration in any way; however, additional selections may be required if the Model structure or any rules have changed.

2.2 Maintaining Custom Servlet Properties

After upgrading to this release, copy or move any custom Oracle Configurator Servlet properties that were previously defined in `jserv.properties` or `zone.properties` to `cz_init.txt`.

For details, see Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4.

Note: Do not copy or move any Java system properties that are defined in `jserv.properties` or `zone.properties` for logging purposes to `cz_init.txt`. If any logging properties are defined in these files, remove them after upgrading and then refer to "Troubleshooting" on page xiv for information about logging.

2.3 Maintaining a DHTML User Interface from a Previous Release

If you have one or more DHTML User Interfaces that you want to continue using after upgrading to this release, consider the following:

- The runtime Oracle Configurator supports DHTML UIs, but you cannot generate a new DHTML UI, or modify an existing DHTML UI, in the HTML-based version of Oracle Configurator Developer.

  To generate or maintain DHTML UIs, you must install the limited edition of Configurator Developer on a client machine. This version of Configurator Developer provides limited functionality: it can be used only to generate, edit, and unit test DHTML UIs. For more information, see the *About Oracle Configurator* documentation for this release on Metalink, Oracle's technical support Web site.

- After upgrading, you must use the HTML-based Configurator Developer to modify the Model structure or any rules, define new rules, or generate a UI that is based on the OA Framework. These tasks are described in the *Oracle Configurator Developer User's Guide*.

  To make any rule or Model structure changes available in a published DHTML UI, perform the following:

  1. Use the HTML-based Configurator Developer to generate logic for the Model.
  2. Use the limited edition of Configurator Developer to refresh the DHTML UI, or generate a new DHTML UI.
  3. Use the limited edition of Configurator Developer to unit test the DHTML UI (optional).
  4. Use the HTML-based Configurator Developer to republish the Model and the updated DHTML UI.

- A Model can have both a DHTML UI and a UI that is generated in the HTML-based Developer. The information in the preceding paragraphs also applies in this scenario.

2.4 Maintaining or Migrating Functional Companions

User Interfaces that you create in the HTML-based version of Configurator Developer are called generated UIs. Generated UIs support all existing Functional Companions except those that interact with the DHTML UI through `IUserInterface` and related Java interfaces.

Note: Oracle recommends that you use Configurator Extensions instead of Functional Companions wherever possible. Circumstances that might require you to maintain Functional Companions are described in this section.

If you want to modify existing Functional Companions and use them with a generated UI, then you must first migrate them to Configurator Extensions. A Configurator Extension that has been migrated from a Functional Companion is supported in both generated and DHTML UIs. For details about Configurator Extensions, see the *Oracle Configurator Extensions and Interface Object Developer's Guide* and the *Oracle Configurator Developer User's Guide*.

- To determine whether you need to migrate Functional Companions, see Section 2.4.1, "Functional Companion Migration Requirements" on page 2-3.
- To understand what happens when Functional Companions are migrated, see Section 2.4.2, "Functional Companion Migration Processing" on page 2-4.
- If you need to run the migration concurrent programs, you must perform the tasks described in Section 2.4.3, "Setup for Migrating Functional Companions" on page 2-6 and use the instructions in the *Oracle Configurator Implementation Guide* to run either the Migrate All Functional Companions or Migrate Functional Companions for a Single Model concurrent program.
- If you need to manually migrate any Functional Companions, as indicated in Table 2–3, " Functional Companion Methods That Are Not Migrated" on page 2-6, see Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
- If you need to retain legacy DHTML UIs that include Functional Companions, and do not need the added functionality provided by Configurator Extensions, see Section 2.4.5, "Maintaining Functional Companions" on page 2-8.
- If you wish to keep using an existing Functional Companion that is *not* already associated with a Model, you can do so by creating Configurator Extension bindings for it, since you cannot use the migration concurrent programs on it. See the *About Oracle Configurator* documentation on Metalink, Oracle's technical support Web site.

2.4.1 Functional Companion Migration Requirements

When deploying Models in this release of Oracle Configurator, your choice about whether or not to migrate existing Functional Companions depends on:

- The need to change Functional Companion behavior
- The type of User Interface used by the Model with which the Functional Companion is associated

These factors are summarized in Table 2–1 on page 2-3.

Table 2–1 Scenarios for Migration of Functional Companions

Change Functional Companion Behavior?	User Interface Used	Migration Needed?	Migration Result
No	Legacy (DHTML) or generated in the HTML-based version of Configurator Developer	No	n/a[1]
Yes	Legacy (DHTML) or generated in the HTML-based version of Configurator Developer	Yes	Functional Companions replaced with Configurator Extensions

[1] Existing Functional Companions still operate.

Functional Companions that interact with a DHTML User Interface (using `IUserInterfaceEventListener.handleUserInterfaceEvent()`) are not supported in generated UIs, and cannot be migrated.

If your Functional Companion code uses `Configuration.getUserInterface()` to test for the existence of a user interface (for example, so that it can work with batch validation, which has no user interface), it should continue to do so. If the test occurs when the configuration is rendered in a UI generated in the HTML-based version of Oracle Configurator Developer, then it always returns null.

Certain Functional Companions that are registered through runtime events must be manually migrated. See Table 2–3, " Functional Companion Methods That Are Not Migrated" on page 2-6.

2.4.2 Functional Companion Migration Processing

Functional Companion migration transforms Functional Companion Rules into Configurator Extension Rules, transforming the data that associates existing Java classes with Models. Migration does not transform the Java classes, although you may need to do that task yourself, depending on the factors described in this document.

Table 2–2, " Migration of Functional Companion Methods to Configurator Extension Event Bindings" on page 2-5 shows the correspondences between the Java methods that are used by Functional Companions and the configuration events used by the Configurator Extension Rules produced by migration.

Other relevant facts about migration are as follows:

- You cannot migrate an individual Functional Companion Rule. You can migrate all the Functional Companion Rules for a specified Model (and its referenced children), or for all Models in the database.
- The migration process creates, where appropriate, a **binding** between the specified Functional Companion method and the corresponding configuration event. The migration process also creates any **argument** bindings required by the event, and sets the event scope for the bindings.
- When migration completes successfully, the Functional Companion association data is logically deleted from the database.
- If a method in a Functional Companion has no arguments, then after migration there are no argument bindings in the resulting Configurator Extension Rule.
- There are certain configuration events used by Configurator Extensions that do not correspond with Functional Companion methods. Consequently, these events do not appear in Table 2–2 on page 2-5.
- Functional Companions in published Models can not be migrated. Only Functional Companions in source Models can be migrated.
- There are certain Functional Companion methods that cannot be migrated by the concurrent programs, and require manual migration. These are included in Table 2–3, " Functional Companion Methods That Are Not Migrated" on page 2-6.
- The Functional Companion methods listed in Table 2–2 on page 2-5 override the methods of the classes `FunctionalCompanion` or `AutoFunctionalCompanion`.
- The `autoConfigure()` method is migrated to an onCommand event binding in which the command name is generated by concatenating the internal Rule ID generated by Oracle Configurator for the new Configurator Extension Rule with a

string indicating the Functional Companion type (_AC for "autoConfigure"). Example: 1000001_AC.

The generateOutput(HttpServletResponse) method is migrated the same way, except that the concatenated string for "generateOutput" is _GO. Example: 1000002_GO.

When you generate a User Interface for your Model, the generated command name is applied to the button that is created, as both its associated action and its default caption. For details on generating User Interfaces and modifying buttons, see the *Oracle Configurator Developer User's Guide*.

Table 2–2 Migration of Functional Companion Methods to Configurator Extension Event Bindings

Functional Companion Method	Configuration Event for Binding	When Migrated
initialize (IRuntimeNode, String, String, int)	postCXInit	Always. This binding is always created, even if the Functional Companion does not already implement initialize().
terminate()	preCXTerminate	If the Functional Companion extends the class FunctionalCompanion or AutoFunctionalCompanion.
autoConfigure()	onCommand (Rule ID + _AC)	If **Auto-Configuration** was selected as the Type of the Functional Companion and if the Functional Companion extends the class FunctionalCompanion or AutoFunctionalCompanion.
generateOutput (HttpServletResponse)	onCommand (Rule ID + _GO)	If **Output** was selected as the Type of the Functional Companion and if the Functional Companion extends the class FunctionalCompanion or AutoFunctionalCompanion.
validate()	onConfigValidate	If **Validation** was selected as the Type of the Functional Companion and if the Functional Companion extends the class FunctionalCompanion or AutoFunctionalCompanion.
validateEligibleTarget (Component)	onValidateEligibleTarget	If the Functional Companion extends the class FunctionalCompanion or AutoFunctionalCompanion.
afterSave()	postConfigSave	If the Functional Companion extends AutoFunctionalCompanion
onLoad()	onInstanceLoad	If the Functional Companion extends AutoFunctionalCompanion
onNew()	postConfigNew	If the Functional Companion extends AutoFunctionalCompanion
onRestore()	postConfigRestore	If the Functional Companion extends AutoFunctionalCompanion
onSave()	preConfigSave	If the Functional Companion extends AutoFunctionalCompanion
onSummary()	preConfigSummary	If the Functional Companion extends AutoFunctionalCompanion
postLoad()	postInstanceLoad	If the Functional Companion extends AutoFunctionalCompanion

There are a number of methods that might be used in Functional Companions that are not migrated by the concurrent programs described in this section. These methods are listed in Table 2–3, " Functional Companion Methods That Are Not Migrated" on page 2-6.

Table 2–3 Functional Companion Methods That Are Not Migrated

Method	Reason Not Migrated
`Functional Companion.generateOutput()`	This deprecated method requires a thick client connection, which is not compatible with Oracle Applications Release 11*i*.
`ICompSetEventListener. notifyComponentAdded(Component)`	Requires manual migration. See Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
`ICompSetEventListener. notifyComponentDeleted(Component)`	Requires manual migration. See Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
`IConfigEventListener. notifyComponentAdded(Component)`	Requires manual migration. See Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
`IConfigEventListener. notifyComponentDeleted(Component)`	Requires manual migration. See Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
`IConfigurationEventListener. eventOccured(ConfiguationEvent)`	Only supported for legacy DHTML user interfaces to Oracle Configurator.
`IConnectionEventListener. notifyConnectionAssigned(Connector)`	Requires manual migration. See Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
`IConnectionEventListener. notifyConnectionUnassigned(Connector)`	Requires manual migration. See Section 2.4.4, "Manual Migration of Certain Functional Companion Methods" on page 2-7.
`IUserInterfaceEventListener. handleUserInterfaceEvent(Event)`	Only supported for Legacy DHTML user interfaces to Oracle Configurator.

2.4.3 Setup for Migrating Functional Companions

You must perform certain setup tasks when migrating Functional Companions.

- Before you run the concurrent programs to migrate Functional Companions, you must put the Java classes that implemented the Functional Companions (which might be in Java archive files in JAR or Zip format) into the class path of the Concurrent Manager. Assuming as an example that your Java classes are in an Java archive file named `fc.jar`, use the following procedure:

  1. The file `fc.jar` must be in a directory that is accessible from the computer on which your Concurrent Manager is started. If it is not, then add the full class path of `fc.jar` (including `fc.jar`) to the variable `AF_CLASSPATH` in the file `$APPL_TOP/admin/adovar.env`.
  2. Initialize your environment again, by logging out and logging in.
  3. Restart your Concurrent Managers.

- Before you run the concurrent programs to migrate Functional Companions, you must ensure that `servlet.jar` is in the system classpath.
- After you run the concurrent programs to migrate Functional Companions, remove the Java classes that implemented the Functional Companions from the class path of your Web server. Then create Configurator Extension Archives for the Java classes, as described in the *Oracle Configurator Developer User's Guide*. If you leave the Java classes in the class path, the runtime Oracle Configurator will use those versions instead of the versions in the Configurator Extension Archives.

2.4.4 Manual Migration of Certain Functional Companion Methods

This section describes in general terms how to manually migrate the methods that require manual migration (see Table 2–3, " Functional Companion Methods That Are Not Migrated" on page 2-6). These methods cannot be migrated by the migration concurrent programs because they require identification of entities that exist only at runtime.

You must adapt the following instructions to the specific characteristics of your Functional Companions.

Example Functional Companion Requiring Manual Migration

A Functional Companion might use the methods in question as shown in the fragment in Example 2–1, "Functional Companion Code Before Manual Migration" on page 2-7.

Example 2–1 Functional Companion Code Before Manual Migration

```
public class MyCompanion extends FunctionalCompanion implements
IConfigEventListener {
    public void initialize(){
        m_config.addConfigEventListener(this); // m_config is a Configuration
    }

    public void notifyComponentAdded(Component comp){ // FC method
        // do something
    }

    public void notifyComponentDeleted(Component comp){ // FC method
        // do something
    }
}
```

General Procedure for Manual Migration

The following steps describe the general procedure for manually migrating this Functional Companion to a Configurator Extension.

1. Rewrite, compile, and archive the Java code so that it resembles the fragment in Example 2–2, "Functional Companion Code After Manual Migration" on page 2-7, which is compatible with Configurator Extensions. For details, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

Example 2–2 Functional Companion Code After Manual Migration

```
public class MyExtension {  // no need to extend CIO classes

  public void componentAdded(ComponentSet set, Component comp){ // custom method
     // do something
    }
```

```
  public void componentDeleted(ComponentSet set, Component comp){ // custom method
      // do something
    }
}
```

For details on the remaining steps, see the *Oracle Configurator Developer User's Guide*.

2. In Oracle Configurator Developer, create a Configurator Extension Archive for the Java archive file containing your revised code.
3. In Oracle Configurator Developer, create a Configurator Extension Rule, specifying your custom class (`MyExtension`).

   Set the **Base Node** of the Rule to the desired node.

   Set the Instantiation Scope of the Rule to With Model Node Instance Set.
4. In this Configurator Extension Rule, create an event binding for each method of your Java class. The following table shows a choice of events that is appropriate to the Java methods in the example. You must choose events that are appropriate to your own case.

Method Name	Event	Event Parameters
`componentAdded(set, comp)`	`postInstanceAdd`	`compSet` `instance`
`componentDeleted(set, comp)`	`postInstanceDelete`	`compSet` `instance`

For each binding, bind the arguments of the Java method to parameters that are required for the event. The following table shows a choice of argument bindings that is appropriate to both of the Java methods in the example. You must choose bindings that are appropriate to your own case.

Argument Type	Argument Name	Argument Specification	Binding
ComponentSet	`set`	Event Parameter	`compSet`
Component	`comp`	Event Parameter	`instance`

2.4.5 Maintaining Functional Companions

If you need to retain legacy DHTML UIs that include Functional Companions, and do not need the added functionality provided by Configurator Extensions, then you do not need to migrate your Functional Companions. Instead, you can maintain the definitions of your Functional Companions in Oracle Configurator Developer, under the following conditions:

- For a legacy Model that has been upgraded to this release, each unmigrated Functional Companion is presented by default on a read-only page in the Rules area of the Workbench Oracle Configurator Developer. This page enables you to examine the definition of the Functional Companion rule. To migrate the Functional Companion, you must run the Migrate Functional Companions for a Single Model concurrent program, as described in Section 2.4, "Maintaining or Migrating Functional Companions" on page 2-2.

- If you set the profile option CZ: Enable Creation of Functional Companions to `Yes`, then each unmigrated Functional Companion is presented on a page in the Rules area of the Workbench in Oracle Configurator Developer that enables you to edit its definition. (For details about this profile option, see Section 1.3.1.11 on page 1-8.) Functional Companion rules have the same attributes in this release as in the previous release: Name, Description, Type (Validation, Auto-Configuration, Output, and Event-Driven), and Program String.
  - If you set the profile option to `Yes`, then you can also create new Functional Companion rules, by clicking the Create icon in the Rules area of the Workbench, then choosing **Functional Companion**. You should only create such rules in an upgraded Model, to be included in a legacy DHTML UI. You should not create Functional Companion rules in a generated UI created with the HTML-based version of Oracle Configurator Developer.
  - See Section 1.3.1.11, "CZ: Enable Creation of Functional Companions" on page 1-8 for details about the profile option that enables editing or creation of Functional Companion rules.

2.5 Maintaining Configuration Attributes

To use configuration attributes with Release 11.5.10, you may need to use Configurator Extensions instead of Functional Companions. For details about configuration attributes, see the *Oracle Configurator Methodologies* documentation.

3

Oracle Configurator Servlet Considerations

This chapter tells you how to verify that the Oracle Configurator Servlet (OC Servlet) is set up correctly and describes all customizable OC Servlet properties.

This chapter contains the following sections:

3.1 Introduction

To view a User Interface in a runtime Oracle Configurator, you must have the Oracle Configurator Servlet (OC Servlet) installed on your internet server.

Installing the OC Servlet includes:

1. Using Oracle Rapid Install to install Oracle Configurator, Oracle Configurator Developer, and Oracle Internet Application Server (*i*AS). Installing *i*AS also installs the Apache Web server and supporting software.

   See Section 1.2, "Oracle Rapid Install" on page 1-1.

2. Configuring Apache and JServ to work with the OC Servlet by verifying (and modifying, if necessary) the Apache configuration files.

   For details, see Section 3.3, "Verifying Apache and JServ Setup" on page 3-2.

3. Verifying or modifying Java system properties for the OC Servlet.

   See Section 3.4, "Oracle Configurator Servlet Properties" on page 3-7.

You may want to consult the *Apache 1.3 User's Guide* and Apache Web site (`http://java.apache.org`) when installing or configuring the OC Servlet.

> **Note:** These instructions assume that you are installing on the Solaris ™ Operating Environment platform, unless noted otherwise.

Note: Run the Oracle Configurator Servlet under the production version of the latest version of JDK 1.4 for your platform. The production version runs significantly faster than the reference version. (Oracle Rapid Install provides JDK version 1.4.2) See Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4 for details on verifying this setting.

3.2 Servlet Properties and Legacy User Interfaces

Previous versions of Oracle Configurator Developer generated either DHTML or Java applet-style UIs. The current version of Configurator Developer generates User Interfaces that are based on the OA Framework, and cannot be used to generate or modify DHTML or Java applet UIs. If you are upgrading to the current version of Configurator Developer and need to create or maintain DHTML or Java applet UIs, see Section 2.3, "Maintaining a DHTML User Interface from a Previous Release" on page 2-2.

This release of Oracle Configurator supports both new and legacy UIs. For details about the Oracle Configurator servlet properties that support UIs created in the current version of Configurator Developer, see Section 3.4.1, "Descriptions of Oracle Configurator Servlet Properties" on page 3-7.

For a description of servlet properties that support DHTML and Java applet UIs, see:

- The *Oracle Configurator Installation Guide* specific to the Oracle Configurator release from which you recently upgraded.

  For example, if you upgraded from release 11.5.8, refer to the release 11.5.8 version of the *Oracle Configurator Installation Guide*.

- The *About Oracle Configurator* documentation for this release

3.3 Verifying Apache and JServ Setup

After you have installed Apache and its supporting software with Oracle Rapid Install, verify that the server is set up correctly. To do this, enter a command with the following structure in a Web browser:

URL of the Servlet?test=version

For example:

```
http://www.mysite.com:60/configurator/oracle/apps/cz/servlet/UiServlet?test=versio
n
```

The result should be the build and schema version of Oracle Configurator running on the server.

If the build and schema version does *not* appear, refer to Oracle *i*AS documentation for details about troubleshooting your Apache setup.

- For details on Apache configuration files, consult the Apache documentation (at `http://java.apache.org`).
- Verify that the mount points (such as ApJServGroupMount and ApJServGroup) are correct. These are defined when you run Oracle Rapid Install.
- You must log in as the owner of the Apache files in order to modify these files.
- Refer to the following sections for information about each configuration file:

- Section 3.3.1, "Verifying httpd.conf" on page 3-4
- Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4

- For a description of the OC Servlet properties, see Section 3.4 on page 3-7.

In this chapter, various textual placeholders are used. Table 3–1 lists the placeholders that may require some explanation (the names of the other placeholders should be self-explanatory).

Table 3–1 Textual Placeholders for Configuration Files

Placeholder	Example Values	Comment
ias_install	/d01/oracle/viscomn/util/apache/1.3.9/Apache	The directory in which you install *i*AS, using Oracle Rapid Install.
apache_install	*ias_install*/Apache (for example, /d01/oracle/viscomn/util/apache/1.3.9/Apache/)	The directory in which you install Apache, as part of *i*AS, using Oracle Rapid Install.
jserv_install	*ias_install*/Jserv (for example, /d01/oracle/viscomn/util/apache/1.3.9/Apache/Jserv)	The directory in which you install JServ, as part of *i*AS, using Oracle Rapid Install.
hostname	*www.mysite.com*	The name of the host machine.
portnum	*8802*	The port number used by the Apache listener, which is specified by *Port* in *httpd.conf*.
html_vpath	OA_HTML	When running under Oracle Applications, the location pointed to by $APPL_TOP/html.
	apache_install/htdocs/html	Otherwise, a directory located under /htdocs, which is specified by DocumentRoot in httpd.conf.
media_vpath	*OA_MEDIA*	When running under Oracle Applications, the location pointed to by $APPL_TOP/media.
	apache_install/htdocs/media	Otherwise, a directory located under /htdocs, which is specified by DocumentRoot in httpd.conf.
servlet_vpath	This should always be: configurator	A mounting location specified by ApJServGroupMount in jserv.conf.

In the case of a placeholder that refers to an environment variable, the configuration file should contain the actual value of the environment variable, not the variable itself. For example, for a placeholder such as:

local_value_of_$APPL_TOP

your configuration file should contain:

/d01/oracle/visappl

rather than:

$APPL_TOP

3.3.1 Verifying httpd.conf

By default, Oracle Rapid Install places `httpd.conf` in *`apache_install`*`/conf`.

Oracle Rapid Install sets the following parameters to point to the appropriate locations:

```
ServerRoot "apache_install"

DocumentRoot "apache_install/htdocs"

Alias /icons/ "apache_install/icons/"

Alias /OA_HTML/ "$APPL_TOP/html"

Alias /OA_MEDIA/ "$APPL_TOP/html"
```

Most of these parameters have corresponding `<Directory>` entries that should also be verified.

Note the use of trailing slash characters added to certain parameters.

You can ignore any settings of the aliases `/html/` and `/media/`. They are not used by the OC Servlet.

Verify that Apache's listening port is one that is not being used on the server machine:

```
Port portnum
```

At the very end of `httpd.conf`, verify that there is a line that points to the location of the JServ configuration file `jserv.conf`, which is located in *`jserv_install`*`/etc`. For example:

```
Include jserv_install/etc/jserv.conf
```

Verify that the Timeout parameter is set to a minimum of 1800 seconds (30 minutes).

```
Timeout 1800
```

3.3.2 Verifying jserv.properties and cz_init.txt

The only Oracle Configurator-specific property in `jserv.properties` that is read by the OC Servlet is `cz_properties_file`. The other properties in `jserv.properties` are provided only to support legacy User Interfaces (DHTML and Java applet) that were created in previous versions of Oracle Configurator Developer.

The property `cz_properties_file` identifies the location of the file `cz_init.txt`. By default, `cz_init.txt` is empty. If you want to customize any servlet properties (for example, by modifying properties that have default values), you must define them in `cz_init.txt`. For details, see Section 3.3.2.2, "Syntax and Context for Setting Parameters" on page 3-5.

If you recently upgraded to this release of Oracle Configurator, you must copy (or move) any custom properties that were previously defined in `jserv.properties` or `zone.properties` to `cz_init.txt`. The OC Servlet ignores any properties that are defined in `jserv.properties` and `zone.properties`.

You can leave obsolete OC Servlet parameters in a configuration file without triggering an error. You can also set unimplemented parameters without triggering an error, if you observe the rules for syntax. You may want to do this for testing purposes, to observe which parameters are passed to the OC Servlet.

Note: The `cz_init.txt` file is intended only for properties that are specific to Oracle Configurator. Defining non-Oracle Configurator, properties in `cz_init.txt` (such as those used for logging) may produce unintended results at runtime. All properties used by the Apache Web server, such as those included in Section 3.3.2.1, "Java Requirements" on page 3-5, are defined in `jserv.properties`.

By default, Oracle Rapid Install places `jserv.properties` in *`jserv_install`*`/etc`.

The file `cz_init.txt` is stored in your Web server's `Jserv` directory. For example:

```
.../Apache/Jserv/etc/cz_init.txt
```

The properties are set on the Java Virtual Machine (JVM) that runs the JServ servlet engine so they are available to the OC Servlet when it starts up.

3.3.2.1 Java Requirements

Verify that the JServ engine is using the production version (as opposed to the reference version) of the JDK. For example:

```
wrapper.bin=/local/java/jdk1.4p/bin/java
```

Oracle strongly recommends that you run Java 1.4 or higher. (Oracle Rapid Install installs this version by default.) However, if you are using an earlier version (such as 1.2.2), ensure that the JServ engine is using native (rather than "green") threads. Version 1.3 uses native threads by default. You can verify that the JServ engine is using native threads by checking that the following parameter is the first parameter specified in `jserv.properties`:

```
wrapper.bin.parameters=-native
```

On platforms that are running Java with green threads and have no native thread implementation, specify the `Xss8m` option so that you use an 8 MB native stack. To specify this option, verify that the following parameter is the first parameter specified in `jserv.properties`:

```
wrapper.bin.parameters=-Xss8m
```

Note: If your operating system is HP-UX, you must use JDK version 1.2.2.08 or higher.

Verify that a single entry such as the following is added to set the maximum heap size:

```
wrapper.bin.parameters=-Xmx1500m
```

Oracle recommends at least 600MB, but the required heap size may vary depending on your configuration.

For more information about heap size allocation, see the *Oracle Configurator Performance Guide*.

3.3.2.2 Syntax and Context for Setting Parameters

You set Apache parameters that determine any customized properties for the OC Servlet in the file `cz_init.txt` (in other words, properties that are used only by

Oracle Configurator). You set these as system execution parameters for the JServ JVM, using the following syntax:

```
wrapper.bin.parameters=property_name=property_value
```

Example:

```
wrapper.bin.parameters=-Dcz.uimanager.logpath=/d01/oracle/viscomn/util/apache/1.3.
9/Apache/Jserv/logs/
```

This syntax is like using the `-D` option of the Java interpreter:

```
java -Dproperty_name=property_value
```

The following example uses the property cz.activemodel, which is described on page 3-8:

```
wrapper.bin.parameters=-Dcz.activemodel=/nolp|/nodp|
```

3.3.2.3 Verifying the Classpath for the Oracle Configurator Servlet

Oracle Rapid Install may set additional classpath entries. Only the entries affecting the OC Servlet are described here.

The following entries are required for the operation of the Apache Web server and the JServ engine:

```
wrapper.classpath=jserv_install/libexec/ApacheJServ.jar

wrapper.classpath=ias_install/Jsdk/lib/jsdk.jar
```

Verify that the `wrapper.env` section specifies Oracle Configurator's load library path:

```
wrapper.env=library_path=local_value_of_$CZ_TOP/bin:$ORACLE_HOME/lib
```

The `library_path` must include the directory in which shared object files are available. The name of `library_path` and the shared object files vary by platform. See Table 3–2 on page 3-6 for details.

Table 3–2 Library Path Variable Names and File Names by Operating System

Operating System	Variable Name	Library File
Solaris ™ Operating Environment or Linux	LD_LIBRARY_PATH	libczlce.so
HP	SHLIB_PATH	libczlce.sl
AIX	LIBPATH	libczlce.sl
NT	PATH	czlce.dll

For example, if the operating system is the Solaris ™ Operating Environment and the database name is "db1":

```
wrapper.env=LD_LIBRARY_
PATH=/u012/oracle/crp1appl/cz/11.5.0/bin:/d2/11i/db1/db1ora/8.0.6/
```

> **Note:** After running Oracle Rapid Install, the shared object files may need to be relinked with `adrelink`.

Verify that the port number matches the ApJServDefaultPort that is set in `jserv.conf`:

```
port=portnum_jserv
```

Verify that the following parameters point to *`jserv_install`*:

```
root.properties=jserv_install/etc/zone.properties

log.file=jserv_install/logs/jserv.log
```

3.3.2.4 Functional Companions, Configurator Extensions, and Return URL Servlets

Functional Companion classes and Return URL servlets must be located in the classpath of the OC Servlet. For example, if you install your Functional Companion classes in *`jserv_install`*`/configurator`, then add the following parameter:

```
wrapper.classpath=jserv_install/configurator
```

See the *Oracle Configurator Implementation Guide* for information on Return URL servlets.

The Java classes that implement the behavior of Configurator Extensions are located in Configurator Extension Archives, which are created in Oracle Configurator Developer and are stored in the database. The runtime Oracle Configurator loads Configurator Extensions from the database, so it is not necessary to install them relative to the OC Servlet.

See the *Oracle Configurator Developer User's Guide* for information about creating Configurator Extension Archives. See the *Oracle Configurator Extensions and Interface Object Developer's Guide* for information about creating the Java classes that implement the behavior of Configurator Extensions.

Note: When you modify and recompile a Functional Companion, you must restart the OC Servlet to load the new class file. It is not necessary to restart the OC Servlet when you modify Configurator Extensions, because they are loaded from the database.

3.4 Oracle Configurator Servlet Properties

OC Servlet properties are passed as Java system properties to the JVM in which the process for the OC Servlet is running.

You can modify the default runtime behaviors produced by the OC Servlet properties described in this section by defining them in `cz_init.txt` and specifying custom values. For details, see Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4.

3.4.1 Descriptions of Oracle Configurator Servlet Properties

This section lists the OC Servlet properties and their default values, which are included in the software code. To modify a property's default behavior, define the property in `cz_init.txt` using the syntax listed in this section, and specify a custom value. For more information about `cz_init.txt`, see Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4.

The properties of the OC Servlet for which you can set Apache configuration parameters are listed in Table 3–3 on page 3-8.

Table 3–3 Properties for the Oracle Configurator Servlet

Property Name	Description
cz.activemodel	on page 3-8
cz.runtime.use_dedicated_jvm	on page 3-9
cz.uiserver.applet_client_poll_wait	on page 3-9
cz.uiserver.check_heartbeat_timeout	on page 3-10
cz.uiserver.heartbeat_interval	on page 3-11
cz.uiserver.poll_timeout_applet	on page 3-12
cz.uiservlet.dio_share	on page 3-12
cz.uiservlet.pre_load_filename	on page 3-13
cz.uiservlet.versionfuncsavail	on page 3-13

cz.activemodel

This property primarily controls whether prices and available to promise (ATP) information are displayed at runtime. It also controls other runtime behavior as described in Table 3–4.

The default pricing behavior for the OC Servlet is that all price fetching is turned off; that is, as though the property were defined as follows:

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

To turn on pricing or enable the other runtime behavior described in Table 3–4, define this property in `cz_init.txt`. For more information, see Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4.

The syntax for this property is:

```
cz.activemodel=/switch|
```

Where *`switch`* is one of the values listed in Table 3–4.

Table 3–4 Switch Values for cz.activemodel

Switch	Effect
lp	Fetch List Prices from the database.
dp	Fetch Discounted (selling) Prices from the database.
atp	Fetch ATP data from the database.
nolp	Do not fetch List Prices from the database.
nodp	Do not fetch Discounted (selling) Prices from the database.
noatp	Do not fetch ATP data from the database.
nofc	Disable Configurator Extensions and Functional Companions. See the *Oracle Configurator Extensions and Interface Object Developer's Guide* for details.

You can set more than one switch. The syntax for this setting is:

```
cz.activemodel=/switch1|/switch2|
```

Note that each switch is separated by the pipe character (|), and that the pipe character is required at the end of the property setting.

If you set contradictory switches, then only the first switch is respected. Example:

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

If a Configurator Developer user creates a UI control to display prices in a generated UI, but one of the switches that disable fetching has been set, the end user will receive an error message when pricing is requested in a runtime Oracle Configurator. The UI controls that Configurator Developer generates to display and update prices are automatically hidden at runtime when pricing is disabled. For details, see the *Oracle Configurator Developer User's Guide*.

Examples of Setting Pricing Switches

The following table lists examples of setting pricing switches, and their effects.

Table 3–5 Pricing Switch Settings

Setting	Effect
`cz.activemodel=/lp|`	List Prices will be fetched and displayed.
`cz.activemodel=/nolp|`	List Pricing is turned off. List Prices will not be fetched or displayed
`cz.activemodel=/lp|/dp|`	List Prices and Discounted Prices will both be fetched and displayed.
`cz.activemodel=/nolp|/nodp|`	No List Prices or Discounted Prices will be fetched.

For more information about pricing and ATP, see the *Oracle Configurator Implementation Guide*.

cz.runtime.use_dedicated_jvm

Set this property to `true` if you want to run Oracle Configurator on a different server than the one on which Oracle Applications is running.

Do not define this property or set it to `false` if you want Oracle Configurator to run on the same server as Oracle Applications. This is the default behavior.

For more information, see Section 1.3.1.1, "BOM: Configurator URL of UI Manager" on page 1-6.

Syntax:

```
cz.runtime.use.dedicated_jvm=[true|false]
```

Example:

```
cz.runtime.use.dedicated_jvm=true
```

cz.uiserver.applet_client_poll_wait

If your operating system is Macintosh version 9*x* running mrj 2.2*x* and you have implemented Secure Sockets Layer (SSL), this property sets the length of time that the Applet session "sleeps" between polls to the server, allowing the client session (Web browser) a free connection to the server. If your operating system is not the specific Macintosh version listed above or you have not implemented SSL, the OC Servlet ignores this property.

Background: In Macintosh version 9*x*, mrj 2.2*x*, legacy UIs (that is Java Applet and DHTML) and User Interfaces generated in Configurator Developer do not maintain separate connections when polling the server. This adversely affects the performance of the UI and may even cause it to fail.

See "The Heartbeat Mechanism and Guided Selling" on page 3-10 for additional information.

The suggested range is 5,000 to 30,000 milliseconds. The default value is 15,000 milliseconds.

> **Note:** Setting this property to a small value (such as 5,000) may affect runtime performance. Specifying a large value (such as 30,000) improves runtime performance, but increases the time required to return control to the OM Sales Order window when the Oracle Configurator session ends.

Syntax:

```
cz.uiserver.applet_client_poll_wait=milliseconds
```

Example:

```
cz.uiserver.applet_client_poll_wait=20000
```

cz.uiserver.check_heartbeat_timeout

This property controls the timeout for the UI Server's checking of "heartbeat" events. (See "The Heartbeat Mechanism and Guided Selling" on page 3-10 for a description of heartbeat events.) If the UI Server doesn't receive any heartbeats from the Web browser after this time value, then the UI Server will end the configuration session and send a "terminate" message back to the Applet client. The default value for this property is 30,000 milliseconds.

Set this property to a value that is 3 times the value of `cz.uiserver.heartbeat_interval` and `cz.uiserver.poll_timeout` (these properties should have the same value). For example, if these properties are set to `20000`, then set `cz.uiserver.check_heartbeat_timeout` to `60000`.

If loading a large configuration model, set this property to a value that is approximately how long it takes to load the Model. For example, if the configuration model takes 60 seconds to load, set this property to approximately 60000 milliseconds.

Syntax:

```
cz.uiserver.check_heartbeat_timeout=milliseconds
```

Example:

```
cz.uiserver.check_heartbeat_timeout=60000
```

The Heartbeat Mechanism and Guided Selling

When a Forms-based Oracle Applications product launches a User Interface that was generated in Configurator Developer, an Oracle Configurator Applet client runs but is not visible to the end user. The Applet client cannot determine the status of the end user's client Web browser, so it does not know if the browser has experienced an error or the end user closed it prematurely.

To address this problem, Oracle Configurator uses a "heartbeat" mechanism, in which:

1. The client session (the Web browser) sends heartbeat events to a session that is running in the UI Server. Continued heartbeats indicate that the client is still "alive". A cessation of heartbeats indicate that the client has terminated. This cessation is detected by the session that is running on the UI Server.
2. The Applet client polls an Applet session running in the UI Server to check whether the UI Server has received a termination message from the client session.
3. If the frequency of heartbeats received by the client session are less than the amount specified by cz.uiserver.heartbeat_interval, then the UI Server sends the termination message to the Applet session, which is being polled by the Applet client running under Order Management.

Table 3–6 lists the servlet properties that control the operation of the heartbeat mechanism.

Table 3–6 Heartbeat Mechanism Properties

Property	Page Reference
cz.uiserver.check_heartbeat_timeout	on page 3-10
cz.uiserver.heartbeat_interval	on page 3-11
cz.uiserver.poll_timeout_applet	on page 3-12

The value for all of these heartbeat parameters must be greater than zero, and must be less than the timeout value for the Web listener. (For Apache, this listener timeout value is specified by the setting for `Timeout` in `httpd.conf`.)

To use guided buying or selling in Order Management, you must also ensure the JServ engine uses native threads. See Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4.

cz.uiserver.database_poll_timeout

This property specifies the amount of time that the server session waits before contacting the database to indicate that the configuration session is still active (this is known as "polling" the database). This system property is used only by HTML User Interfaces; in other words, UIs created in release 11.5.10 of Oracle Configurator Developer.

You may want to increase the default value if your environment has minimal system resources (for example, few database connections), since doing so will cause the server to contact the database less frequently, thereby reducing the possibility that the configuration session will expire prematurely.

The default value is 5000 milliseconds (5 seconds).

Syntax:

```
cz.uiserver.database_poll_timeout=value
```

Example:

```
cz.uiserver.database_poll_timeout=15000
```

cz.uiserver.heartbeat_interval

This property controls the frequency at which the heartbeat is sent from the client browser to the UI Server. The default value is 10000 milliseconds. The suggested range

for this property is between 10,000 and 60,000. This property and `cz.uiserver.poll_timeout_applet` should be set to the same value.

See "The Heartbeat Mechanism and Guided Selling" on page 3-10 for background.

Syntax:

```
cz.uiserver.heartbeat_interval=milliseconds
```

Example:

```
cz.uiserver.heartbeat_interval=10000
```

cz.uiserver.poll_timeout_applet

This property sets the time after which the UI Server's Applet session tells the Applet client to poll back, to check whether the UI Server session was terminated. The default value is 10,000 milliseconds. The suggested range for this property is between 10,000 and 60,000. This property and `cz.uiserver.heartbeat_interval` should be set to the same value.

See "The Heartbeat Mechanism and Guided Selling" on page 3-10 for background.

Syntax:

```
cz.uiserver.poll_timeout_applet=milliseconds
```

Example:

```
cz.uiserver.poll_timeout_applet=10000
```

cz.uiservlet.dio_share

Controls whether the UI Server running inside the OC Servlet shares (caches) the Model in the DIO between configuration sessions.

The default value is `true`, which enables sharing of the cached Model. Sharing the cached Model improves the loading performance of sessions after the first one for a given Model, but requires that the OC Servlet be restarted in order for the runtime Oracle Configurator to reflect recent changes to generated logic. However, configuration sessions started with the Test button in Oracle Configurator Developer ignore the cached Model and fetch the latest Model from the database, thus reflecting changes to generated logic. This setting provides a convenience for Model developers, while providing efficiency for runtime users. As a general rule, you should not change the default value (`true`).

Setting this property to `false` disables sharing of the cached Model for *all* configuration sessions on the same OC Servlet.

Syntax:

```
cz.uiservlet.dio_share=[true|false]
```

Example:

```
cz.uiservlet.dio_share=true
```

> **Note:** This property provides a development convenience, when you disable Model caching. However, this convenience counteracts the performance enhancement derived by preloading and caching a Model at servlet startup, by using cz.uiservlet.pre_load_filename.

cz.uiservlet.pre_load_filename

This property specifies an absolute path to the file containing an initialization message for the OC Servlet. This file is read if you are preloading the servlet. In order to preload a servlet with Apache, you must specify its class name as the value of the parameter `servlets.startup` in the file `cz_init.txt`. For more information about `cz_init.txt`, see Section 3.3.2, "Verifying jserv.properties and cz_init.txt" on page 3-4.

Example for setting `servlets.startup`:

```
servlets.startup=oracle.apps.cz.servlet.UiServlet
```

Syntax for this property:

```
cz.uiservlet.pre_load_filename=absolute_path_to_init_file
```

The contents of `init_file` is a valid Oracle Configurator initialization message, the construction of which is described in detail in the *Oracle Configurator Implementation Guide*. Make sure that each initialization message in *`init_file`* is completely on a single line, with no line breaks in the message text. There can be multiple initialization messages in the file, but each message must be on its own line.

Example for an initialization message:

```
<initialize>
<param name="database_id">dbc_filename</param>
<param name="gwyuid">applsyspub/pub</param>
<param name="user"apps</param><param name="pwd">apps</param>
<param name="ui_type">JRAD</param>
<param name="context_org_id">5</param>
<param name="model_id">1234</param>
<param name="calling_application_id">671</param>
</initialize>
```

Example for setting this property:

```
cz.uiservlet.pre_load_filename=/home/apache/init_msg.txt
```

> **Note:** This property provides a performance enhancement by caching a Model at servlet startup. However, this enhancement is counteracted if you disable Model caching by setting cz.uiservlet.dio_share to `false`.

cz.uiservlet.versionfuncsavail

Use this property to determine whether the servlet responds to the `test=version` message entered in a Web browser. The default value is `true`.

For details, see:

- Section 3.3, "Verifying Apache and JServ Setup" on page 3-2
- Section 4.3, "Checking the Response of the UI Servlet" on page 4-2

Syntax:

```
cz.uiservlet.versionFuncsAvail=[true|false]
```

Example:

```
cz.uiservlet.versionFuncsAvail=true
```

4

Troubleshooting Servlet Installation

This chapter provides suggestions for resolving problems that may arise when installing the Oracle Configurator Servlet. This installation is described in Chapter 3, "Oracle Configurator Servlet Considerations".

This chapter contains the following sections:

4.1 Introduction

Oracle Configurator Developer and Oracle Configurator use the standard Oracle Applications methods of logging to analyze and debug both development and runtime issues. These methods include setting various profile options and Java system properties to enable logging and specify the desired level of detail you want to record.

For more information about logging, see:

- The *Oracle Applications System Administrator's Guide*. This document provides descriptions of the Oracle Applications Manager UI screens that allow System Administrators to set up logging profiles, review Java system properties, search for log messages, and so on.
- The *Oracle Applications Supportability Guide*. This document includes logging guidelines for both System Administrators and developers, and related topics.
- The Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1). This document provides troubleshooting information that is specific to the OA Framework.

4.2 Before You Begin

Before troubleshooting OC Servlet installation, be sure that:

- The Oracle Applications profile option BOM: Configurator URL of UI Manager points to the location of the Oracle Configurator Servlet.

  See "BOM: Configurator URL of UI Manager" on page 1-6.

- You have set your virtual paths correctly.

  See Table 3–1 on page 3-3.

- Your executable path includes the Shared Object files, (.so or .dll). A symptom of this problem might be an error message starting with a line similar to the following:

  ```
  java.lang.UnsatisfiedLinkError: no czjni in shared library path
  ```

4.3 Checking the Response of the UI Servlet

What you are checking

Does the UI Servlet respond to a test message?

The test

Invoke this URL in a Web browser:

http://hostname:portnum/configurator/oracle.apps.cz.servlet.UiServlet?test=test_version

where *hostname* is the name of your internet server, *portnum* is the port number for your Web listener, and *configurator* is the virtual path that you set up when installing the servlet. If the servlet is installed correctly and running, it should produce an HTML page that prints the current build version of Oracle Configurator and the expected version for the CZ schema.

> **Note:** The property cz.uiservlet.versionfuncsavail determines whether you can test the response of the servlet using a test string. For details, see "cz.uiservlet.versionfuncsavail" on page 3-13.

Example:

```
http://www.mysite.com:8802/configurator/oracle.apps.cz.servlet.UiServlet?test=version
```

produces a result like the following:

```
Using configuration software build: 11.5.10.22.8
Expecting schema: 22a
```

If the test fails

- Enable logging, or increase the level of detail that Configurator Developer records by changing settings available by clicking the Diagnostics global link. Look in the log file to see which classes it loads, and from which JAR files. There may be a message indicating that some classes failed to load. It is probably the case that there is a JAR file in the list that is not in the path specified or that there was an error in specifying its name.

  For details about the logging options that are available via the Diagnostics global link, see the Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

- There may be a basic problem with your Apache configuration. See Section 4.5, "Checking the Operation of the Apache Internet Server" on page 4-4.

- Verify all configurator classes are in the java directory in your class path.

- The JServ engine may not have been started up. Start the JServ engine, and then perform the test again.

4.4 Checking Your Model in the Runtime Oracle Configurator

What you are checking

Does your configuration model behave as you expect in the runtime Oracle Configurator?

The test

You can launch a generated User Interface from Oracle Configurator Developer by clicking the Test Model button in the Structure, Rules, or User Interface area of the Workbench. For more information, see the chapter on unit testing in the *Oracle Configurator Developer User's Guide*.

Alternatively, you can test the behavior of the runtime Oracle Configurator by creating a test page that substitutes for your host application.

To do this:

1. Create an HTML test page that posts an initialization message to the UI Servlet.

   See the chapter on session initialization in the *Oracle Configurator Implementation Guide* for an explanation of the OC initialization message.

   See Example 4–1, and the *Oracle Configurator Implementation Guide* for examples of simple test pages.

Example 4–1 Test Page for Invoking the JRAD Runtime Oracle Configurator

```
<html>
<head>
<title>Minimal Configurator Test</title>
</head>
<body>
<form action="http://www.mysite.com:8802/OA_HTML/CZInitialize.jsp" method="post">

<input type="hidden" name="XMLmsg" value=
'<initialize>
  <param name="database_id">serv01_sid02</param>
  <param name="user">operations</param>
  <param name="pwd">welcome</param>
  <param name="calling_application_id">708</param>
  <param name="responsibility_id">22713</param>
  <param name="ui_type">JRAD</param>
  <param name="ui_def_id">3120</param>
</initialize>'>
<p>Click the button to configure the model...
<input type="submit" value="Configure">
</form>
</body>
</html>
```

2. Ensure that you have the necessary database connectivity, and that your UI Servlet is installed and configured correctly.

   For details, see Section 3.3, "Verifying Apache and JServ Setup" on page 3-2.

3. Test the runtime Oracle Configurator by opening the test page.

Your default Web browser should open and contain the specified configuration model and User Interface. If you used Example 4–1, click the button to display the specified User Interface.

4.5 Checking the Operation of the Apache Internet Server

Test 1 - What you are checking

Does your Apache internet server respond at all?

The test

1. Compile the code in Example 4–2 into the file Hello.class in your servlets directory.
2. In a Web browser, invoke this URL:

```
http://hostname:portnum/servlet_vpath/Hello
```

where *hostname* is the server that you have installed on, *portnum* is the port number configured for the HTTP listener, and *servlet_vpath* is the mounting location specified by ApJServGroupMount in the file `Jserv.conf` (this is usually `configurator`). For example:

```
http://www.mysite.com:10130/configurator/Hello
```

3. The browser should display the HTML message written by your test class.

Example 4–2 Hello.java Test Class

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * This is a simple example of an HTTP Servlet.  It responds to the GET
 * and HEAD methods of the HTTP protocol.
 */
public class Hello extends HttpServlet
{
    /**
     * Handle the GET and HEAD methods by building a simple Web page.
     * HEAD is just like GET, except that the server returns only the
     * headers (including content length) not the body we write.
     */
    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException
        {
            PrintWriter out;
            String title = "Example Apache JServ Servlet";

            // set content type and other response header fields first
            response.setContentType("text/html");

            // then write the data of the response
            out = response.getWriter();

            out.println("<HTML><HEAD><TITLE>");
            out.println(title);
            out.println("</TITLE></HEAD><BODY bgcolor=\"#FFFFFF\">");
```

```
            out.println("<H1>" + title + "</H1>");
            out.println("<H2> Congratulations, ApacheJServ is working!<br>");
            out.println("</BODY></HTML>");
            out.close();
        }
}
```

If the test fails

Check that the internet server installation, the port number, and that the server you chose is available on the right network. Sometimes this can be a server name problem. To get around that problem, refer to the server by its IP address.

Test 2 - What you are checking

Is the hostname and port valid?

The Test

In a Web browser, invoke this URL:

`http://`*`hostname`*`:`*`portnum`*`/`

If the test is successful, the "Oracle Applications Rapid Install Portal" page appears. This indicates that Apache is working and that the hostname and port are valid. If this test works, but the first test did not, then there could be problems with the JServ/JVM configuration. Check the mount point (ApJServGroupMount directive) in `jserv.conf` and look for errors in the mod_jserv_log and jserv_log files. Additionally, confirm that Java is setup correctly (change directory to the Java interpreter path specified in `jserv.properties` and enter "java -version").

If this test returns an error, there are problems with Apache on the machine. Check the error_log, make sure `httpd.conf` (or `httpds.conf`) is set up properly, and make sure the hostname is a resolvable DNS entry.

Test 3 - What you are checking

Does the /OA_HTML/ alias exist and, if so, is it valid?

The Test

In a Web browser, invoke this URL:

`http://`*`hostname`*`:`*`portnum`*`/OA_HTML/czblank.jsp`

Verify that the page is displayed (it will look like HTML source code). If this test fails, be sure that:

- The /OA_HTML/ alias is set up correctly (in `httpd.conf`, `httpds.conf`, `oracle_apache.conf`, or `apps.conf`)
- There is no `OA_HTML` directory under the DocumentRoot (`httpd.conf` or `httpds.conf`)

Check Apache Configuration Files

If these tests do not resolve the problem, check the following Apache/JServ configuration files.

❑ In `JServ.conf`, check the settings of the following parameters and modify them as required:

- **ApJServGroupMount**: This parameter specifies the mount points for servlets zones. Be sure the following entry exists:

  ```
  ApJServGroupMount /configurator balance://GroupName/root
  ```

  where *GroupName* is the name of the group defined in jserv.conf file. (A group is a set of processes across which request traffic is distributed for load-balancing purposes.)

  See Section 3.3, "Verifying Apache and JServ Setup" on page 3-2.

❑ In the `jserv.properties` file, check the following and make changes as required:

- The path of the Java interpreter (java.exe or jre.exe) - Near the top of the file (just under "# The Java Virtual Machine interpreter") there is a line that begins with `wrapper.bin`. This must point to the full path of the Java executable (Java or JRE). This is usually apache_install/jdk/bin/java, where apache_install is replaced by the actual path.

  For example:

  ```
  wrapper.bin=/u09/oracle/test1ora/iAS/Apache/jdk/bin/java
  ```

  Note that the JRE or JDK (Java) can be used in many cases, but in other cases only the JDK can be used (for example, Oracle CRM). Rapid Install installs the JDK on the Solaris ™ Operating Environment, but it is a manual process on other platforms.

❑ The Shared Library Path variable must point to the directory that contains the Configurator shared library(s). This path is always $CZ_TOP/bin.

The shared library path variable name is platform-dependent. Table 3–2 lists the library path name for each operating system.

❑ Verify the classpath for the OC Servlet. See Section 3.3.2.3, "Verifying the Classpath for the Oracle Configurator Servlet" on page 3-6.

Glossary

This glossary contains definitions that you may need while working with Oracle Configurator.

API

Application Programming Interface

applet

A Java application running inside a Web browser. *See also* **Java** and **servlet**.

Archive Path

The ordered sequence of **Configurator Extension Archive**s for a **Model** that determines which **Java class**es are loaded for **Configurator Extension**s and in what order.

argument

A data value or object that is passed to a method or a **Java class** so that the method can operate.

ATO

Assemble to Order

ATP

Available to Promise

base node

The **node** in a **Model** that is associated with a **Configurator Extension** Rule. Used to determine the **event** scope for a **Configurator Extension**.

bill of material

A list of Items associated with a parent Item, such as an assembly, and information about how each Item relates to that parent Item.

Bills of Material

The application in Oracle Applications in which you define a **bill of material**.

binding

Part of a **Configurator Extension** Rule that associates a specified event with a chosen **method** of a **Java class**. *See also* **event**.

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**.

Boolean Feature

An **element** of a **component** in the **Model** that has two **option**s: true or false.

bug

See **defect**.

build

A specific **instance** of an application during its construction. A build must have an install program early in the project so that application **implementer**s can **unit test** their latest work in the context of the entire available application.

CDL

See **Constraint Definition Language**.

CIO

See **Oracle Configuration Interface Object (CIO)**.

command event

An **event** that is defined by a character string, which is considered the command for which **listener**s are listening.

Comparison Rule

An **Oracle Configurator Developer** rule type that establishes a relationship to determine the selection state of a logical **Item** (Option, Boolean Feature, or List-of-Options Feature) based on a comparison of two numeric values (numeric **Feature**s, **Total**s, **Resource**s, **Option** counts, or numeric constants). The numeric

values being compared can be computed or they can be discrete intervals in a continuous numeric input.

Compatibility Rule

An **Oracle Configurator Developer** rule type that establishes a relationship among **Feature**s in the Model to control the allowable combinations of **Option**s. *See also*, **Property-based Compatibility Rule**.

Compatibility Table

A kind of Explicit Compatibility Rule. For example, a type of compatibility relationship where the allowable combination of **Option**s are explicitly enumerated.

component

A piece of something or a configurable element in a **model** such as a **BOM Model**, **Model**, or **Component**.

Component

An element of the **model structure**, typically containing **Feature**s, that is configurable and instantiable. An **Oracle Configurator Developer** node type that represents a configurable element of a **Model**. Corresponds to one UI screen of selections in a runtime **Oracle Configurator**.

Component Set

An element of the **Model** that contains a number of instantiated **Component**s of the same type, where each Component of the set is independently configured.

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.

configuration

A specific set of specifications for a product, resulting from selections made in a runtime **configurator**.

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 engine

The part of the runtime **Oracle Configurator** that uses **configuration rule**s to validate a **configuration**. Compare **generated logic**.

Configuration Interface Object

See **Oracle Configuration Interface Object (CIO)**.

configuration model

Represents all possible configurations of the available **option**s, and consists of **model structure** and **rules**. It also commonly includes **User Interface** definitions and **Configurator Extension**s. A configuration model is usually accessed in a **runtime Oracle Configurator window**. *See also* **model**.

configuration rule

A **Logic Rule**, **Compatibility Rule**, **Comparison Rule**, **Numeric Rule**, **Design Chart**, **Statement Rule**, or **Configurator Extension** rule available in **Oracle Configurator Developer** for defining **configuration**s. *See also* **rules**.

configuration session

The time from launching or invoking to exiting **Oracle Configurator**, during which **end user**s 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 host application so **end user**s can make selections resulting in valid **configuration**s. *Compare* **Oracle Configurator**.

Configurator Extension

An extension to the **configuration model** beyond what can be implemented in Configurator Developer.

A type of **configuration rule** that associates a **node**, **Java class**, and event **binding** so that the rule operates when an **event** occurs during a **configuration session**.

A **Java class** that provides methods that can be used to perform configuration actions.

Configurator Extension Archive

An **object** in the **Repository** that stores one or more compiled **Java class**es that implement **Configurator Extension**s.

connectivity

The connection between client and database that allows data communication.

The connection across components of a model that allows modeling such products as networks and material processing systems.

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**.

Constraint Definition Language

A language for entering **configuration rule**s as text rather than assembling them interactively in Oracle Configurator Developer. CDL can express more complex constraining relationships than interactively defined configuration rules can.

Container Model

A type of **BOM Model** that you import from Oracle **Bills of Material** into **Oracle Configurator Developer** to create configuration models containing **connectivity** and trackable components. Configurations created from Container Models can be tracked and updated in Oracle Install Base

Contributes to

A relation used to create a specific type of **Numeric Rule** that accumulates a total value. *See also* **Total**.

Consumes from

A relation used to create a specific type of **Numeric Rule** that decrements a total value, such as specifying the quantity of a **Resource** used.

count

The number or quantity of something, such as selected **option**s. *Compare* **instance**.

CTO

Configure to Order

customer

The person for whom products are configured by **end user**s of the **Oracle Configurator** or other **ERP** and CRM applications. Also the end users themselves directly accessing **Oracle Configurator** in a Web store or kiosk.

customer requirements

The needs of the customer that serve as the basis for determining the configuration of products, **system**s, and services. Also called needs assessment. *See* **guided buying or selling**.

CZ

The product shortname for **Oracle Configurator** in Oracle Applications.

CZ schema

The implementation version of the standard runtime **Oracle Configurator** data-warehousing schema that manages data for the **configuration model**. The implementation schema includes all the data required for the **runtime** system, as well as specific tables used during the construction of the **configurator**.

data import

Populating the **CZ schema** with enterprise data from **ERP** or legacy systems via **import tables**.

data source

A programmatic reference to a database. Referred to by a data source name (DSN).

DBMS

Database Management System

default

A predefined value. In a **configuration**, the automatic selection of an **option** based on the **preselection** rules or the selection of another option.

Defaults relation

An **Oracle Configurator Developer** Logic Rule relation that determines the logic state of **Feature**s or **Option**s in a default relation to other Features and Options. For example, if A Defaults B, and you select A, B becomes Logic True (selected) if it is available (not Logic False).

defect

A failure in a product to satisfy the **user**s' requirements. Defects are prioritized as critical, major, or minor, and fixes range from corrections or workarounds to enhancements. Also known as a bug.

Design Chart

An **Oracle Configurator Developer** rule type for defining advanced Explicit Compatibilities interactively in a table view.

developer

The person who uses **Oracle Configurator Developer** to create a **configurator**. *See also* **implementer** and **user**.

Developer

The tool (**Oracle Configurator Developer**) used to create **configuration model**s.

DHTML

Dynamic Hypertext Markup Language

discontinued item

A discontinued item is one that exists in an installed configuration of a component (as recorded in Oracle Install Base), but has been removed from the instance of the component being reconfigured, either by deletion or by deselection.

element

Any entity within a **model**, such as **Option**s, **Total**s, **Resource**s, UI controls, and **component**s.

end user

The ultimate user of the runtime **Oracle Configurator**. The types of end users vary by project but may include salespeople or distributors, administrative office staff, marketing personnel, order entry personnel, product engineers, or customers directly accessing the application via a Web browser or kiosk. *Compare* **user**.

enterprise

The **system**s and **resource**s of a business.

environment

The arena in which software tools are used, such as operating system, applications, and **server** processes.

ERP

Enterprise Resource Planning. A software system and process that provides automation for the customer's back-room operations, including order processing.

event

An action or condition that occurs in a **configuration session** and can be detected by a **listener**. Example events are a change in the value of a **node**, the creation of a component **instance**, or the saving of a **configuration**. The part of **model structure** inside which a **listener** listens for an event is called the event **binding** scope. The part of model structure that is the source of an event is called the event execution scope. *See also* **command event**.

Excludes relation

An **Oracle Configurator Developer Logic Rule** type that determines the logic state of **Feature**s or **Option**s in an excluding relation to other Features and Options. For example, if A Excludes B, and if you select A, B becomes Logic False, since it is not allowed when A is true (either User or Logic True). If you deselect A (set to User

False), there is no effect on B, meaning it could be User or Logic True, User or Logic False, or **Unknown**. *See* **Negates relation**.

feature

A characteristic of something, or a configurable element of a **component** at **runtime**.

Feature

An element of the **model structure**. Features can either have a value (numeric or Boolean) or enumerated **Option**s.

functional specification

Document describing the functionality of the application based on **user** requirements.

generated logic

The compiled structure and rules of a **configuration model** that is loaded into memory on the Web server at **configuration session** initialization and used by the **Oracle Configurator engine** to validate runtime selections. The logic must be generated either in **Oracle Configurator Developer** or programmatically in order to access the configuration model at **runtime**.

guided buying or selling

Needs assessment questions in the **runtime** UI to guide and facilitate the configuration process. Also, the **model structure** that defines these questions. Typically, guided selling questions trigger **configuration rule** that automatically select some product **option**s and exclude others based on the **end user**'s responses.

host application

An application within which **Oracle Configurator** is embedded as integrated functionality, such as Order Management or *i*Store.

HTML

Hypertext Markup Language

implementation

The stage in a project between defining the problem by selecting a configuration technology vendor, such as Oracle, and deploying the completed configuration application. The implementation stage includes gathering requirements, defining test cases, designing the application, constructing and testing the application, and delivering it to **end user**s. *See also* **developer** and **user**.

implementer

The person who uses **Oracle Configurator Developer** to build the **model structure**, **rules**, and UI customizations that make up a **runtime** Oracle Configurator. Commonly also responsible for enabling the integration of **Oracle Configurator** in a **host application**.

Implies relation

An **Oracle Configurator Developer Logic Rule** type that determines the logic state of **Feature**s or **Option**s in an implied relation to other Features and Options. For example, if A Implies B, and you select A, B becomes Logic True. If you deselect A (set to User False), there is no effect on B, meaning it could be User or Logic True, User or Logic False, or **Unknown**. *See* **Requires relation**.

import server

A database **instance** that serves as a source of data for **Oracle Configurator**'s Populate, Refresh, and Synchronization concurrent processes. The import server is sometimes referred to as the remote server.

import tables

Tables mirroring the CZ schemaItem Master structure, but without integrity constraints. Import tables allow batch population of the CZ schema's Item Master. Import tables also store extractions from Oracle Applications or **legacy data** that create, update, or delete records in the CZ schema **Item Master**.

initialization message

The **XML** message sent from a **host application** to the **Oracle Configurator Servlet**, containing data needed to initialize the runtime Oracle Configurator. *See also* **termination message**.

Instance

An **Oracle Configurator Developer** attribute of a **component's node** that specifies a minimum and maximum value. *See also* **instance**.

instance

A **runtime** occurrence of a **component** in a configuration. *See also* **instantiate**. *Compare* **count**.

Also, the memory and processes of a database.

instantiate

To create an instance of something. Commonly, to create an **instance** of a **component** in the runtime **user interface** of a **configuration model**.

integration

The process of combining multiple software **component**s and making them work together.

integration testing

Testing the interaction among software programs that have been integrated into an application or **system**. Also called system testing. *Compare* **unit test**.

item

A product or part of a product that is in inventory and can be delivered to customers.

Item

A Model or part of a Model that is defined in the **Item Master**. Also data defined in Oracle Inventory.

Item Master

Data stored to structure the Model. Data in the **CZ schema** Item Master is either entered manually in **Oracle Configurator Developer** or imported from Oracle Applications or a legacy system.

Item Type

Data used to classify the Items in the Item Master. Item Catalogs imported from Oracle Inventory are Item Types in **Oracle Configurator Developer**.

Java

An object-oriented programming language commonly used in internet applications, where Java applications run inside Web browsers and **server**s. Used to implement the behavior of **Configurator Extension**s. *See also* **applet** and **servlet**.

Java class

The compiled version of a **Java** source code file. The **method**s of a Java class are used to implement the behavior of **Configurator Extension**s.

JavaServer Pages

Web pages that combine static presentation elements with dynamic content that is rendered by Java **servlet**s.

JSP

See **JavaServer Pages**.

legacy data

Data that cannot be imported without creating custom extraction programs.

listener

A class in the **CIO** that detects the occurrence of specified **event**s in a **configuration session**.

load

Storing the **configuration model** data in the **Oracle Configurator Servlet** on the Web server. Also, the time it takes to initialize and display a configuration model if it is not preloaded.

The burden of transactions on a **system**, commonly caused by the ratio of **user** connections to CPUs or available memory.

log file

A file containing errors, warnings, and other information that is output by the running application.

Logic Rule

An **Oracle Configurator Developer** rule type that expresses constraint among model elements in terms of logic relationships. Logic Rules directly or indirectly set the logical state (User or Logic True, User or Logic False, or **Unknown**) of **Feature**s and **Option**s in the Model.

There are four primary Logic Rule relations: Implies, Requires, Excludes, and Negates. Each of these rules takes a list of Features or Options as operands. *See also* **Implies relation**, **Requires relation**, **Excludes relation**, and **Negates relation**.

maintainability

The characteristic of a product or process to allow straightforward **maintenance**, alteration, and extension. Maintainability must be built into the product or process from inception.

maintenance

The effort of keeping a **system** running once it has been deployed, through **defect** fixes, procedure changes, infrastructure adjustments, data replication schedules, and so on.

Metalink

Oracle's technical support Web site at:

http://www.oracle.com/support/metalink/

method

A function that is defined in a **Java class**. Methods perform some action and often accept parameters.

Model

The entire hierarchical "tree" view of all the data required for **configuration**s, including **model structure**, variables such as **Resource**s and **Total**s, and elements in support of intermediary rules. Includes both imported **BOM Model**s and Models created in Configurator Developer. May consist of BOM Option Classes and BOM Standard Items.

model

A generic term for data representing products. A model contains **element**s that correspond to **item**s. Elements may be **component**s of other objects used to define products. A **configuration model** is a specific kind of model whose elements can be configured by accessing an **Oracle Configurator window**.

model-driven UI

The graphical views of the **model structure** and **rules** generated by **Oracle Configurator Developer** to present **end user**s with interactive product selection based on **configuration model**s.

model structure

Hierarchical "tree" view of data composed of **element**s (**Model**s, **Component**s, **Feature**s, **Option**s, **BOM Model**s, **BOM Option Class node**s, **BOM Standard Item node**s, **Resource**s, and **Total**s). May include reusable **component**s (**Reference**s).

Negates relation

A type of **Oracle Configurator Developer Logic Rule** type that determines the logic state of **Feature**s or **Option**s in a negating relation to other Features and Options. For example, if one **option** in the relationship is selected, the other option must be Logic False (not selected). Similarly, if you deselect one option in the relationship, the other option must be Logic True (selected). *See* **Excludes relation**.

node

The icon or location in a **Model** tree in **Oracle Configurator Developer** that represents a **Component**, **Feature**, **Option** or variable (**Total** or **Resource**), **Connector**, **Reference**, **BOM Model**, **BOM Option Class node**, or **BOM Standard Item node**.

Numeric Rule

An **Oracle Configurator Developer** rule type that expresses constraint among model elements in terms of numeric relationships. *See also*, **Contributes to** and **Consumes from**.

object

Entities in **Oracle Configurator Developer**, such as **Model**s, Usages, Properties, Effectivity Sets, UI Templates, and so on. *See also* **element**.

OC

See **Oracle Configurator**.

OCD

See **Oracle Configurator Developer**.

option

A logical selection made in the Model Debugger or a runtime Oracle Configurator by the **end user** or a rule when configuring a **component**.

Option

An element of the **Model**. A choice for the value of an enumerated **Feature**.

Oracle Configuration Interface Object (CIO)

A **server** in the **runtime** application that creates and manages the interface between the client (usually a **user interface**) and the underlying representation of **model structure** and **rules** in the **generated logic**.

The CIO is the **API** that supports creating and navigating the Model, querying and modifying selection states, and saving and restoring **configuration**s.

Oracle Configurator

The product consisting of development tools and **runtime** applications such as the **CZ schema**, **Oracle Configurator Developer**, and runtime Oracle Configurator. Also the runtime Oracle Configurator variously packaged for use in networked or Web deployments.

Oracle Configurator architecture

The three-tier **runtime** architecture consists of the **User Interface**, the **generated logic**, and the **CZ schema**. The application development architecture consists of **Oracle Configurator Developer** and the CZ schema, with test instances of a runtime **Oracle Configurator**.

Oracle Configurator Developer

The suite of tools in the **Oracle Configurator** product for constructing and maintaining **configurator**s.

Oracle Configurator engine

The part of the **Oracle Configurator** product that validates runtime selections. *See also* **generated logic**.

Oracle Configurator schema

See **CZ schema**.

Oracle Configurator Servlet

A **Java** servlet that participates in rendering Legacy user interfaces for **Oracle Configurator**.

Oracle Configurator window

The **user interface** that is launched by accessing a **configuration model** and used by **end user**s to make the selections of a **configuration**.

performance

The operation of a product, measured in throughput and other data.

Populator

An entity in **Oracle Configurator Developer** that creates **Component**, **Feature**, and **Option node**s from information in the **Item Master**.

preselection

The default state in a **configurator** that defines an initial selection of **Component**s, **Feature**s, and **Option**s for configuration.

A process that is implemented to select the initial element(s) of the **configuration**.

product

Whatever is ordered and delivered to customers, such as the output of having configured something based on a model. Products include intangible entities such as services or contracts.

Property

A named value associated with a **node** in the **Model** or the **Item Master**. A set of Properties may be associated with an Item Type. After importing a BOM Model, Oracle Inventory Catalog Descriptive Elements are Properties in **Oracle Configurator Developer**.

Property-based Compatibility Rule

An **Oracle Configurator Developer** Compatibility Rule type that expresses a kind of compatibility relationship where the allowable combinations of **Option**s are specified implicitly by relationships among Property values of the Options.

prototype

A construction technique in which a preliminary version of the application, or part of the application, is built to facilitate **user** feedback, prove feasibility, or examine other implementation issues.

PTO

Pick to Order

publication

A unique deployment of a **configuration model** (and optionally a **user interface**) that enables a developer to control its availability from host applications such as Oracle Order Management or *i*Store. Multiple publications can exist for the same configuration model, but each publication corresponds to only one **Model** and **User Interface**.

publishing

The process of creating a **publication** record in **Oracle Configurator Developer**, which includes specifying applicability parameters to control **runtime** availability and running an Oracle Applications concurrent process to copy data to a specific database.

RDBMS

Relational Database Management System

reference

The ability to reuse an existing **Model** or **Component** within the structure of another Model (for example, as a subassembly).

Reference

An **Oracle Configurator Developer** node type that denotes a **reference** to another **Model**.

Repository

Set of pages in **Oracle Configurator Developer** that contains areas for organizing and maintaining **Model**s and shared **object**s in a single location.

Requires relation

An **Oracle Configurator Developer** Logic Rule relationship that determines the logic state of **Feature**s or **Option**s in a requirement relation to other Features and Options. For example, if A Requires B, and if you select A, B is set to Logic True (selected). Similarly, if you deselect A, B is set to Logic False (deselected). See **Implies relation**.

resource

Staff or equipment available or needed within an enterprise.

Resource

A variable in the **Model** used to keep track of a quantity or supply, such as the amount of memory in a computer. The value of a Resource can be positive or zero, and can have an Initial Value setting. An error message appears at **runtime** when the value of a Resource becomes negative, which indicates it has been over-consumed. Use **Numeric Rule**s to contribute to and consume from a Resource.

Also a specific node type in **Oracle Configurator Developer**. *See also* **node**.

reusable component

See **reference** and **model structure**.

reusability

The extent to and ease with which parts of a **system** can be put to use in other systems.

rules

Also called business rules or **configuration rule**. In the context of Oracle Configurator and **CDL**, a rule is not a "business rule." Constraints applied among elements of the product to ensure that defined relationships are preserved during configuration. Elements of the product are **Component**s, **Feature**s, and **Option**s. Rules express logic, numeric parameters, implicit compatibility, or explicit compatibility. Rules provide **preselection** and **validation** capability in **Oracle Configurator**.

See also **Comparison Rule**, **Compatibility Rule**, **Design Chart**, **Logic Rule** and **Numeric Rule**.

runtime

The environment and context in which applications are run, tested, or used, rather than developed.

The environment in which an **implementer** (tester), **end user**, or **customer** configures a product whose model was developed in **Oracle Configurator Developer**. *See also* **configuration session**.

schema

The tables and objects of a data model that serve a particular product or business process. *See also* **CZ schema**.

server

Centrally located software processes or hardware, shared by client**s**.

servlet

A Java application running inside a Web server. *See also* **Java**, **applet**, and **Oracle Configurator Servlet**.

solution

The deployed **system** as a response to a problem or problems.

SQL

Structured Query Language

Statement Rule

An **Oracle Configurator Developer** rule type defined by using the Oracle Configurator **Constraint Definition Language** (text) rather than interactively assembling the rule's elements.

system

The hardware and software **component**s and infrastructure integrated to satisfy functional and **performance** requirements.

termination message

The **XML** message sent from the **Oracle Configurator Servlet** to a **host application** after a **configuration session**, containing configuration outputs. *See also* **initialization message**.

Total

A variable in the **Model** used to accumulate a numeric total, such as total price or total weight.

Also a specific node type in **Oracle Configurator Developer**. *See also* **node**.

UI

See **User Interface**.

UI Templates

Templates available in **Oracle Configurator Developer** for specifying UI definitions.

Unknown

The logic state that is neither true nor false, but unknown at the time a **configuration session** begins or when a Logic Rule is executed. This logic state is also referred to as Available, especially when considered from the point of view of the **runtime Oracle Configurator end user**.

unit test

Execution of individual routines and modules by the application **implementer** or by an independent test consultant to find and resolve **defect**s in the application. *Compare* **integration testing**.

update

Moving to a new version of something, independent of software release. For instance, moving a production **configurator** to a new version of a **configuration model**, or changing a **configuration** independent of a model **update**.

upgrade

Moving to a new release of **Oracle Configurator** or **Oracle Configurator Developer**.

user

The person using a product or system. Used to describe the person using **Oracle Configurator Developer** tools and methods to build a **runtime Oracle Configurator**. *Compare* **end user**.

User Interface

The part of an **Oracle Configurator** implementation that provides the graphical views necessary to create **configuration**s interactively. A **user interface** is generated from the **model structure**. It interacts with the model definition and the **generated logic** to give **end user**s access to customer requirements gathering, product selection, and any extensions that may have been implemented. *See also* **UI Templates**.

user interface

The visible part of the application, including menus, dialog boxes, and other on-screen elements. The part of a **system** where the **user** interacts with the software. Not necessarily generated in **Oracle Configurator Developer**. *See also* **User Interface**.

user requirements

A description of what the **configurator** is expected to do from the **end user's** perspective.

validation

Tests that ensure that configured **component**s will meet specific criteria set by an enterprise, such as that the components can be ordered or manufactured.

variable

Parts of the **Model** that are represented by **Total**s, **Resource**s, or numeric **Feature**s.

verification

Tests that check whether the result agrees with the specification.

Web

The portion of the Internet that is the World Wide Web.

Workbench

Set of pages in **Oracle Configurator Developer** for creating, editing, and working with **Repository object**s such as **Model**s and **UI Templates**.

XML

Extensible Markup Language, a highly flexible markup language for transferring data between **Web** applications. Used for the **initialization message** and **termination message** of the **Oracle Configurator Servlet**.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W