Oracle® Configurator

Installation Guide

Release 12.1

Part No. E14323-03

August 2010

ORACLE®

Oracle Configurator Installation Guide, Release 12.1

Part No. E14323-03

Primary Author: Mark Sawtelle

Contents

Send Us Your Comments

Oracle Configurator Installation Guide, Release 12.1

Part No. E14323-03

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Oracle E-Business Suite Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

Intended Audience

Welcome to Release 12.1 of 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.

If you are responsible for installing Oracle Configurator, be sure you understand Oracle Applications technology, terminology, and architecture before beginning the installation. For details, see Oracle Applications Documentation Resources, on MetaLink, Oracle's technical support Web site.

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.

See Related Information Sources on page ix for more Oracle E-Business Suite product information.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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.

Structure

1 Introduction

This chapter describes the conventions used in this guide, provides information about troubleshooting issues with Oracle Configurator and Configurator Developer, and tells you how to contact Oracle Support Services.

2 Installing Oracle Configurator

This chapter contains an overview of the Oracle Configurator installation process and things to consider if you are implementing Multiple Language Support (MLS).

3 Upgrading to this Release

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

4 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.

5 Troubleshooting Servlet Installation

This chapter provides suggestions for resolving problems that may arise when installing the Oracle Configurator Servlet. This installation is described in Oracle Configurator Servlet Considerations, page 4-1.

A Profile Options

This appendix describes the profile options that are used by Oracle Configurator and Oracle Configurator Developer.

Common Glossary for Oracle Configurator

Related Information Sources

> **Important:** There is new functionality available for the Runtime Oracle Configurator when using the Fusion Configurator Engine (FCE). The FCE is an alternative to the configuration engine described in this document. For all information about the FCE, see the *Oracle Configurator Fusion Configurator Engine Guide.*.

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

For a full list of documentation resources for Oracle Configurator, see the Oracle Configurator Release Notes for this release.

For a full list of documentation resources for Oracle Applications, see Oracle Applications Documentation Resources, on MetaLink, Oracle's technical support Web site.

Additionally, be sure you are familiar with current release or patch information for Oracle Configurator on MetaLink, Oracle's technical support Web site.

Integration Repository

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

Do Not Use Database Tools to Modify Oracle E-Business Suite Data

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

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

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

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

1 Introduction

This chapter describes the conventions used in this guide, provides information about troubleshooting issues with Oracle Configurator and Configurator Developer, and tells you how to contact Oracle Support Services.

This chapter covers the following topics:

- Conventions
- Product Support

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 guide.

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

Convention	Meaning
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 per cent 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.
&	Indicates a character string (identifier) that can display text dynamically in Configurator Developer or a runtime Oracle Configurator. For example, "&PROPERTY" can be used to dynamically construct and display a Property of a Model structure node.

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:

Example
```
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:
    1. Choose **Product**: Oracle Configurator or Oracle Configurator Developer
    2. 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:

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

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 more information about logging, see:

- The *Oracle E-Business Suite 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 E-Business Suite Developer's Guide*, which includes logging guidelines for both System Administrators and developers, and related topics.
- The *Oracle E-Business Suite Framework Developer's Guide*, which describes the logging options that are available via the Diagnostics global link. This document is available on MetaLink.

2

Installing Oracle Configurator

This chapter contains an overview of the Oracle Configurator installation process and things to consider if you are implementing Multiple Language Support (MLS).

This chapter covers the following topics:

- Overview
- Oracle Rapid Install
- Additional Setup Tasks
- Test Your Oracle Configurator and Oracle Configurator Developer Installation
- Web Browser Requirements
- Installation and Setup Considerations for Multiple Language Support
- Required Patches

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 12 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 Oracle Rapid Install, page 2-2.

If you are upgrading an existing Oracle Configurator installation, see Upgrading to this Release, page 3-1.

Warning: Before beginning to implement your configurator project, attend Oracle Configurator Developer training, read the *Oracle Configurator Developer User's Guide* and review the current release or patch information for Oracle Configurator on MetaLink, Oracle's technical support Web site.

Oracle Rapid Install

Oracle Rapid Install is an automated process that installs Oracle Applications Release 12 and Oracle Application Server 10*g* (Fusion Middleware). Oracle Application Server 10*g* contains 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.

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 Additional Setup Tasks, page 2-2.

For more information about Oracle Rapid Install, see *Oracle E-Business Suite Installation Guide: Using Rapid Install*.

Note: If you are implementing Multiple Language Support (MLS), read Installation and Setup Considerations for Multiple Language Support, page 2-4 before running Rapid Install.

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 Overview of Profile Options, page A-1.

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

   See Verifying Apache and OC4J Setup, page 4-2.

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 E-Business Suite 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 E-Business Suite System Administrator's Guide*.

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

   See Test Your Oracle Configurator and Oracle Configurator Developer Installation, page 2-3.

6. If you are implementing Multiple Language Support (MLS), see Installation and Setup Considerations for Multiple Language Support, page 2-4.

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 Web Browser Requirements, page 2-4.

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 Model logic.

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:

   1. From the Structure, Rules, or User Interface area of the Workbench, click **Test Model**.

   2. Select **User Interface**, and then select the User Interface you want to test.

   3. 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, page 1-2.

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 Application Framework Documentation Resources, Release 12, on MetaLink.

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 E-Business Suite Concepts* for background information on language support in Oracle Applications, including how to select languages, character sets, and territory values.

For information about Rapid Install and National Language Support (NLS), refer to *Oracle E-Business Suite Installation Guide: Using Rapid Install*.

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*.

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.

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.

Required Patches

After installing Oracle Applications, you must run `adpatch` to apply the latest patches to your Oracle Applications Release 12 environment. For the latest required patches, contact Oracle Support or go to MetaLink, Oracle's technical support Web site.

See *Maintaining Oracle E-Business Suite* for information about applying patches.

3

Upgrading to this Release

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

This chapter covers the following topics:

- Introduction
- Maintaining Custom Servlet Properties
- Maintaining or Migrating Functional Companions
- Maintaining Configuration Attributes

Introduction

This chapter contains information about Functional Companions and legacy User Interfaces that you should consider when upgrading from a previous release of Oracle Configurator.

If you are upgrading from a previous version of Oracle Configurator Release 12, run `adpatch` to apply the latest patches to your Oracle Applications Release 12 environment. For the latest required patches, go to MetaLink, Oracle's technical support Web site. See *Maintaining Oracle E-Business Suite* for information about applying these patches.

For additional information about this release of Oracle Configurator and Oracle Configurator Developer, see the current release or patch information for Oracle Configurator on MetaLink.

> **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 since the configuration was saved.

Note: When upgrading from Release 11.5.9 or earlier to Release 12, the table `CZ_EXT_APPLICATIONS` is created. This table is used by Configurator Developer when you publish configuration models. During the upgrade, this table is populated with all of the Oracle Applications products that can host Oracle Configurator. Any other application that has published Models are also added to this table. If you want to publish a Model to an application that does not appear in the list of values for the Applications applicability parameter in Oracle Configurator Developer, then you must add a record to the `CZ_EXT_APPLICATIONS` table using PL/SQL. For more information, see the *Oracle Configurator Implementation Guide*.

Maintaining Custom Servlet Properties

Before upgrading to this release, back up your `cz_init.txt` file and all customizations to Apache configuration files, such as `jserv.properties`, `httpd.conf`, and `zone.properties` (these configuration files are not used by Oracle Application Server). The upgrade process creates an empty `cz_init.txt` file and overwrites the existing version.

After the installation or upgrade is complete, copy any Configurator-related custom properties that were previously defined in `zone.properties` or `jserv.properties` to `cz_init.txt`.

For more information, see Verifying oc4j.properties and cz_init.txt, page 4-3.

Note: Do not copy or move any Java system properties that are defined in `jserv.properties` or `zone.properties` to `cz_init.txt` for logging purposes. Refer to Troubleshooting, page 1-3 for information about logging.

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 interacted with DHTML UIs, since DHTML UIs are no longer supported.

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 generated 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 Functional Companion Migration Requirements, page 3-3.
- To understand what happens when Functional Companions are migrated, see Functional Companion Migration Processing, page 3-4.
- If you need to run the migration concurrent programs, you must perform the tasks described in Setup for Migrating Functional Companions, page 3-9 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 Functional Companion Methods That Are Not Migrated, page 3-8, see Manual Migration of Certain Functional Companion Methods, page 3-9.
- 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 current release or patch information for Oracle Configurator on MetaLink, Oracle's technical support Web site.

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 Scenarios for Migration of Functional Companions, page 3-4.

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	Existing Functional Companions still operate
Yes	Legacy (DHTML) or generated in the HTML-based version of Configurator Developer	Yes	Functional Companions replaced with Configurator Extensions

Functional Companions that interacted with DHTML User Interfaces (using `IUserInterfaceEventListener.handleUserInterfaceEvent()`) are not supported in generated UIs, and cannot be migrated. DHTML UIs themselves are no longer supported.

Certain Functional Companions that are registered through runtime events must be manually migrated. See Functional Companion Methods That Are Not Migrated, page 3-8.

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.

Migration of Functional Companion Methods to Configurator Extension Event Bindings, page 3-6 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 Migration of Functional Companion Methods to Configurator Extension Event Bindings, page 3-6.
- 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 Functional Companion Methods That Are Not Migrated, page 3-8.
- The Functional Companion methods listed in Migration of Functional Companion Methods to Configurator Extension Event Bindings, page 3-6 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*.

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`

Functional Companion Method	Configuration Event for Binding	When Migrated
`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`

- `postConfigInit`
- `postInstanceAdd`
- `postInstanceDelete`
- `postInstanceEditable`
- `postInstanceEditable`
- `postInstanceNonEditable`
- `postInstanceNameChange`
- `postConnect`
- `postDisconnect`
- `postValueChange`
- `preConfigCancel`

- `preConfigDone`
- `preConfigTerminate`

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 Functional Companion Methods That Are Not Migrated, page 3-8.

Functional Companion Methods That Are Not Migrated

Method	**Reason Not Migrated**
Example `Functional Companion.generateOutput()`	This deprecated method requires a thick client connection, which is not compatible with Oracle Applications Release 12.
Example `ICompSetEventListener. notifyComponentAdded(Component)`	Requires manual migration. See Manual Migration of Certain Functional Companion Methods, page 3-9.
Example `ICompSetEventListener. notifyComponentDeleted(Component)`	Requires manual migration. See Manual Migration of Certain Functional Companion Methods, page 3-9.
Example `IConfigEventListener.` `notifyComponentAdded(Component)`	Requires manual migration. See Manual Migration of Certain Functional Companion Methods, page 3-9.
Example `IConfigEventListener. notifyComponentDeleted(Component)`	Requires manual migration. See Manual Migration of Certain Functional Companion Methods, page 3-9.
Example `IConfigurationEventListener. eventOccured(ConfiguationEvent)`	Only supported for legacy DHTML user interfaces to Oracle Configurator. DHTML UIs are no longer supported.
Example `IConnectionEventListener. notifyConnectionAssigned(Connector)`	Requires manual migration. See Manual Migration of Certain Functional Companion Methods, page 3-9.
Example `IConnectionEventListener. notifyConnectionUnassigned(Connector)`	Requires manual migration. See Manual Migration of Certain Functional Companion Methods, page 3-9.

Method	Reason Not Migrated
Example `IUserInterfaceEventListener.handleUserInterfaceEvent(Event)`	Only supported for Legacy DHTML user interfaces to Oracle Configurator. DHTML UIs are no longer supported.

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.

Manual Migration of Certain Functional Companion Methods

This section describes in general terms how to manually migrate the methods that require manual migration (see Functional Companion Methods That Are Not Migrated, page 3-8). 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 Functional Companion Code Before Manual Migration, page 3-10.

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 Functional Companion Code After Manual Migration, page 3-10, which is compatible with Configurator Extensions. For details, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

   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.

5.

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`

Maintaining Functional Companions

If you need to retain the functionality of DHTML UIs that include Functional Companions, then you must migrate your Functional Companions to Configurator Extensions. Functional Companions require DHTML UIs, and DHTML UIs are not supported in this release. The limited edition of Oracle Configurator Developer that is required for creating or modifying DHTML UIs is not available in this release. You should not create Functional Companion rules in a generated UI created with the HTML-based version of Oracle Configurator Developer.

- 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 in 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 Maintaining or Migrating Functional Companions, page 3-2.

- 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 CZ: Enable Creation of Functional Companions, page A-11.) 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 were to set the profile option to `Yes`, then you could also create new Functional Companion rules, by clicking the Create icon in the Rules area of the Workbench, then choosing **Functional Companion**. In previous releases, you could create such rules in an upgraded Model, to be included in a legacy DHTML UI, but DHTML UIs are no longer supported. You should not ever create Functional Companion rules in a generated UI created with the HTML-based version of Oracle Configurator Developer.
  - See CZ: Enable Creation of Functional Companions, page A-11 for details about the profile option that enables editing or creation of Functional Companion rules.

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.

4

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 covers the following topics:

- Introduction
- Servlet Properties and Legacy User Interfaces
- Verifying Apache and OC4J Setup
- Oracle Configurator Servlet Properties

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 Application Server 10*g*. Installing Oracle Application Server 10*g* also installs the Apache Web server and supporting software.

   See Oracle Rapid Install, page 2-2.

2. Verifying OC Servlet setup by reviewing (and modifying, if necessary) the Apache configuration files.

   For details, see Verifying Apache and OC4J Setup, page 4-2.

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

   See Oracle Configurator Servlet Properties, page 4-7.

Refer to Oracle Application Server documentation for additional information about 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 1.5 for your platform. The production version runs significantly faster than the reference version. (Oracle Rapid Install provides JDK version 1.5.) See Verifying oc4j.properties and cz_init.txt, page 4-3 for details on verifying this setting.

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, you cannot use it to create or maintain DHTML or Java applet UIs, but must first upgrade to a previous release that supports maintenance of DHTML UIs or Java applet UIs and then convert them to OA Framework-based UIs.

For details about the Oracle Configurator servlet properties that support UIs created in the current version of Configurator Developer, see Descriptions of Oracle Configurator Servlet Properties, page 4-7.

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*.

- Current release or patch information for Oracle Configurator on MetaLink, Oracle's technical support Web site

Verifying Apache and OC4J Setup

After installing Apache and its supporting software by running 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:

Example

```
URL of the Servlet?test=version
```

Example

```
http://www.mysite.com:60/OA_HTML/configurator/UiServlet?test=version
```

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

Example
```
Using configuration software build: 12.0.27.2
Expecting schema: 27b
```

> **Note:** The property cz.uiservlet.versionfuncsavail must be set to `true` for this test to work properly. For details, see cz.uiservlet.versionfuncsavail, page 4-14.

If the build and schema version does *not* appear, refer to Oracle Application Server documentation for details about troubleshooting server configuration.

- For details about Apache configuration files, refer to Apache documentation at `http://java.apache.org`.
- You must log in as the owner of the configuration files in order to modify them.
- Refer to the following for information about each configuration file:
  - Verifying oc4j.properties and cz_init.txt, page 4-3
  - Documents related to upgrading or installing Oracle Applications. See Oracle Applications Documentation Resources, on MetaLink, Oracle's technical support Web site.
- For a description of the OC Servlet properties, see Oracle Configurator Servlet Properties, page 4-7.

Verifying oc4j.properties and cz_init.txt

Oracle Application Server reads Java system properties from the file `oc4j.properties`. The only Oracle Configurator-specific property in `oc4j.properties` that is read by the OC Servlet is `cz_properties_file`.

The property `cz_properties_file` identifies the location of the file `cz_init.txt`. Define any custom servlet properties in this file (for example, you may want to modify properties that have default values). For details, see Syntax and Context for Setting Parameters, page 4-6.

The other properties in `oc4j.properties` are provided only to support legacy User Interfaces (DHTML and Java applet UIs) that were created in previous versions of Oracle Configurator Developer.

For important information about preserving customized servlet files when installing or upgrading, see Maintaining Custom Servlet Properties, page 3-2.

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 OC4J properties are available as Java system properties and are available to both the OC Servlet and all Oracle Configurator Java classes. By default, Oracle Rapid Install places `oc4j.properties` and `cz_init.txt` in the following directory:

Example

```
/ins/apps/EnvName/ora/10.1.3/j2ee/oacore/config
```

The properties are set on the Java Virtual Machine (JVM) that runs the OC4J container so they are available to the OC Servlet when it starts up.

> **Note:** Oracle Containers for J2EE (OC4J) is the core J2EE runtime component of Oracle Application Server. Refer to Oracle Application Server documentation for details.

Java Requirements

Oracle strongly recommends that you run Java 5 (JDK 1.5) or higher. Oracle Rapid Install installs this version by default.

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

You set custom values for JVM options in the system-specific context file that is created by Oracle Rapid Install. For example:

```
$APPL_TOP/admin/OAInstName.xml
```

where *`OAInstName`* is the name of your Oracle Applications instance.

These options must be set using the context variable `s_oacore_jvm_start_options;` otherwise the values will be lost when you apply patches or upgrade to a new release. The default value of this variable is:

```
<oacore_jvm_start_options oa_var="s_oacore_jvm_start_options">-server
-verbose:gc -Xmx512M -Xms128M
-XX:MaxPermSize=160M -XX:NewRatio=2  -XX:+PrintGCTimeStamps -XX:+UseTLAB
-XX:+UseParallelGC
-XX:ParallelGCThreads=2  -Dcom.sun.management.jmxremote
-Djava.security.policy=$ORACLE_HOME/j2ee/oacore/config/java2.policy
-Djava.awt.headless=true -Dhttp.webdir.enable=false
-Doracle.security.jazn.config=/slot02/appmgr/inst/apps/czr12d19/ora/10.1
.3/j2ee/oacore/config/jazn.xml-Doracle.security.jazn.config=/slot02/appm
gr/inst/apps/czr12d19/ora/10.1.3/j2ee/oacore/config/jazn.xml>
```

The default values for the maximum and initial heap size are `-Xmx512M` and

`-Xms128M`, respectively.

> **Note:** When viewing the context variable `s_oacore_jvm_start_options` in a text editor, its value appears on a single line. The value is shown here on multiple lines to improve readability.

After modifying this context variable, run AutoConfig to populate the system configuration files with new values and make them permanent. For details about AutoConfig, see *Oracle E-Business Suite Concepts*.

Registering a Return URL Servlet

If you have implemented a custom return URL servlet, then perform the following:

> **Note:** This procedure assumes that you are installing a single Java class file for your servlet. If your class is located in a JAR file, then it must be added to the class path for OC4J. See Specifying the Location of Functional Companion Classes, page 4-6 for information on how to add JAR files to the class path for OC4J.

1. Place the compiled Java class file for the servlet under your $JAVA_TOP directory.

   Example: If you created a servlet **`Checkout.java`** in a package **`myorg.myservlets`**, then the location of the class file should be:

   ```
   $JAVA_TOP/myorg/myservlets/Checkout.class
   ```

2. To register an alias name for the return URL servlet you modify the following file:

   ```
   $INST_TOP/ora/10.1.3/j2ee/oacore/application-deployments/oacore/html
   /orion-web.xml
   ```

3. In the `<servlet>` section of `orion-web.xml`, enter a unique alias for the return URL servlet name, and then specify its fully-qualified class name.

   Example:

   ```
   <servlet>
   <servlet-name>Checkout</servlet-name>
   <servlet-class>myorg.myservlets.Checkout</servlet-class>
   </servlet>
   ```

4. In the `<servlet-mapping>` section, define a text pattern that associates your servlet's alias with the URL to the class file's location relative to $JAVA_TOP. ($JAVA_TOP is assumed to be in the runtime class path.)

   Example:

```
<servlet-mapping>
<servlet-name>Checkout</servlet-name>
<url-pattern>/myorg/myservlets/Checkout</url-pattern>
</servlet-mapping>
```

See the *Oracle Configurator Implementation Guide* for more information about implementing the return URL.

Syntax and Context for Setting Parameters

You set Apache parameters that determine any customized OC Servlet properties (that is, properties that are used only by Oracle Configurator) in the file `cz_init.txt`. Set Java system properties using the following syntax:

Example

```
property_name=property_value
```

For example, use the following syntax to set the property cz.uiservlet.pre_load_filename, page 4-12:

Example

```
cz.uiservlet.pre_load_filename=init_msg.txt
```

Specifying the Location of Functional Companion Classes

If you are using legacy Functional Companions, specify the location of required Java archive (JAR) files in `orion-application.xml`. To do this, use one of the following context variables:

- `s_oacore_prepend_classpath`
- `s_oacore_append_classpath`

The file `orion-application.xml` is located in the following directory:

```
$INST_TOP/ora/10.1.3/j2ee/oacore/application-deployments/oacore/
```

The following example shows how to specify a single Functional Companion JAR file in `orion-application.xml`, using the variable `s_oacore_append_classpath`:

Example

```
<oacore_append_classpath oa_var="s_oacore_append_classpath">
FullPathToJARFile
</oacore_append_classpath>
```

where *`FullPathToJARFile`* is the path to your JAR file.

The following example shows how to specify *multiple* JAR files in `orion-application.xml`:

Example

```
<oacore_append_classpath oa_var="s_oacore_append_classpath">
<![CDATA[/FullPathToJARFile/File_1.jar"/> <library path="/
FullPathToJARFile/File_2.jar"/> <library path="/FullPathToJARFile
/File_3.jar]]>
</oacore_append_classpath>
```

To temporarily add a JAR file (such as for testing purposes), edit `orion-application.xml` and add the following:

```
<library path="FullPathToJAROrZIPFile"/>
```

Important: Remember that this change is temporary, as the `library path` entry will be removed when you apply a patch or upgrade to a new release.

Note: 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.

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 Verifying oc4j.properties and cz_init.txt, page 4-3.

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 Verifying oc4j.properties and cz_init.txt, page 4-3.

The properties of the OC Servlet for which you can set Apache configuration parameters are listed in Properties for the Oracle Configurator Servlet, page 4-8.

Properties for the Oracle Configurator Servlet

Property Name
cz.runtime.treebehavior, page 4-8
cz.uiserver.applet_client_poll_wait, page 4-9
cz.uiserver.check_heartbeat_timeout, page 4-9
cz.uiserver.database_poll_timeout, page 4-11
cz.uiserver.heartbeat_interval, page 4-11
cz.uiserver.poll_timeout_applet, page 4-11
cz.uiservlet.dio_share, page 4-12
cz.uiservlet.pre_load_filename, page 4-12
cz.uiservlet.versionfuncsavail, page 4-14
cz.uiserver.lfalse_is_not_available, page 4-14
cz.uiserver.lazyload, page 4-14

cz.runtime.treebehavior

This property controls optional behavior for the Dynamic Tree, which is a navigation aid included in certain User Interfaces that you design with Oracle Configurator Developer.

If you set the value of this property to `ShowLinks`, then the Dynamic Tree Navigation UI displays a pair of links that allow users to expand and collapse the tree. The links are positioned above the tree, and are labeled Expand All and Collapse All.

If this property is not defined, or is set to any other value, the links will not be shown. By default, the links are not displayed.

For more information about the Dynamic Tree Navigation UI templates, see the *Oracle Configurator Developer User's Guide*.

Syntax:

```
cz.runtime.treebehavior=[ShowLinks]
```

Example

```
cz.runtime.treebehavior=ShowLinks
```

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, page 4-10 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:

Example

```
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, page 4-10 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_applet` (these properties should have the same value). For example, if each of 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:

Example
```
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, page 4-11, then the UI Server sends the termination message to the Applet session, which is being polled by the Applet client running under Order Management.

Heartbeat Mechanism Properties, page 4-10 lists the servlet properties that control the operation of the heartbeat mechanism.

Heartbeat Mechanism Properties

Property	Page Reference
Example `cz.uiserver.check_heartbeat_timeout, page 4-9`	cz.uiserver.check_heartbeat_timeout, page 4-9
Example `cz.uiserver.heartbeat_interval, page 4-11`	cz.uiserver.heartbeat_interval, page 4-11

Property	Page Reference
Example `cz.uiserver.poll_timeout_applet, page 4-11`	cz.uiserver.poll_timeout_applet, page 4-11

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.

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 Oracle Configurator Developer release 11.5.10 or later).

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:

Example
```
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, page 4-10 for background.

Syntax:

Example
```
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, page 4-10 for background.

Syntax:

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

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

cz.uiservlet.dio_share

This property controls whether the UI Server running inside the OC Servlet shares configuration model data that is cached in the DIO between configuration sessions.

The default value of this property is `true`, which enables sharing of cached model data. Sharing cached model data improves the loading performance of sessions after the first one for a given configuration model, but requires that the OC Servlet be restarted for the runtime Oracle Configurator to reflect recent changes to generated logic. However, configuration sessions started with the Test Model button in Oracle Configurator Developer always ignore the cached model data and fetch the latest data from the database, thus reflecting changes to generated logic.

The default 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. In other words, a value of `false` counteracts the performance enhancement derived by preloading and caching configuration model data. For more information, see cz.uiservlet.pre_load_filename, page 4-12.

Syntax:

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

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

For information about decaching configuration model data, see the *Oracle Configurator Performance Guide*.

cz.uiservlet.pre_load_filename

Define this property in `cz_init.txt` if you want to improve performance when the OC Servlet starts up by caching configuration model data. This property specifies an absolute path to a file containing an initialization message for the OC Servlet, and this file identifies the configuration model(s) to preload.

This property is not defined by default.

Syntax:

Example

```
cz.uiservlet.pre_load_filename=absolute_path_to_initialization_file
```

Example

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

The contents of your initialization file must be a valid Oracle Configurator initialization message, the construction of which is described in detail in the *Oracle Configurator Implementation Guide*. Make sure that the initialization message appears on a single line with no line breaks in the message text. There can be multiple initialization messages in the file (that is, one message for each Model you want to preload), but each initialization message must be on its own line.

The following is an example of an initialization message (each parameter appears on a separate line here only to improve readability):

Example

```
<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>
```

To preload a servlet with Apache, its class name must also be specified on startup. As of Release 12, all servlets that need to be loaded on startup are specified in the file `orion-web.xml`. The required preload servlet classes for Oracle Configurator are predefined there by default.

To verify that configuration model data is being preloaded:

1. Enable logging.
2. Clear the Oracle Configurator log files.
3. Stop and then restart the server.
4. Launch Oracle Configurator.
5. Check the FND log files and verify that configuration model data is being preloaded.

Note: Preloading configuration models improves runtime performance by caching data when starting the OC Servlet. However, this enhancement is counteracted if you disable caching either by setting cz.uiservlet.dio_share, page 4-12 to `false` or including the parameter `share_dio=false` in your initialization message.

For information about decaching configuration model data, see the *Oracle Configurator Performance Guide*.

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:

- Verifying Apache and OC4J Setup, page 4-2
- Checking the Response of the UI Servlet, page 5-2

Syntax:

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

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

cz.uiserver.lfalse_is_not_available

This property controls the appearance of options at runtime that have been set logically false (LFALSE) by Oracle Configurator. By default, this parameter is set to `false`, and all logically false options appear as if they have never been selected (that is, as if their logic state were actually UNKNOWN). Set this parameter to `true` if you want Oracle Configurator to indicate logically false options using an icon. The default status indicator icons are described in the *Oracle Configurator Developer User's Guide*.

> **Note:** For best runtime performance, the default value (false) is recommended if you have large, complex Models with many options. Smaller, less complex models with fewer options usually perform better with this property set to true.

Syntax:

Example
```
cz.uiserver.lfalse_is_not_available=[true|false]
```

Example
```
cz.uiserver.lfalse_is_not_available=false
```

cz.uiserver.lazyload

This property controls how UI pages are loaded when an end user begins a configuration session. Choose a value for this property based on how many child (referenced) Models are required in the configuration, as well as application page sizes and usage.

For example:

- Accept the default value of 0 (zero) if you want to load only the top level nodes in the Model (for example, the root Model node and all of its child Component nodes).

  This value is recommended if, for example, the Model contains many child Models that are required in the configuration (that is, the Instances Minimum value for multiple referenced Models is set to 1 or more in Configurator Developer).

- Set this property to 1 if you want to load all of the top level nodes in the Model, and all required child Models.

- Set this property to 2 if you want to load all of the top level nodes in the Model, all required child Models, *and* all of the UI pages for these nodes.

- Set this property to 3 if you want to load all nodes, UI pages, and Model data.

If you set this property to a low value (such as 0 or 1), the initialization of the configuration session is faster, but the first visit to any page is potentially slower (this is because the OC Servlet loads a UI page only when the end user navigates to it). However, after a page is viewed once, it loads quickly whenever the end user returns to the page. This behavior may be more desirable if end users are likely to navigate to a specific page multiple times during a configuration session.

If you set this property to a higher value (such as 2 or 3), the initialization of the configuration session is slower, but the first visit to any UI page is potentially faster. This is because all of the UI pages are loaded when the configuration session is initialized. The pages also load quickly when an end user navigates to the page. Subsequent visits to a page are also fast. If an end user is likely to visit each page only once, and a fast initial load is not critically important, it may be appropriate to set this property to either 2 or 3.

Syntax:

Example
```
cz.uiserver.lazyload=[0|1|2|3]
```

Example
```
cz.uiserver.lazyload=0
```

> **Note:** Consider the behavior of this property with regard to the behavior provided by cz.uiservlet.dio_share, page 4-12 and cz.uiservlet.pre_load_filename, page 4-12.

5

Troubleshooting Servlet Installation

This chapter provides suggestions for resolving problems that may arise when installing the Oracle Configurator Servlet. This installation is described in Oracle Configurator Servlet Considerations, page 4-1.

This chapter covers the following topics:

- Introduction
- Before You Begin
- Checking the Response of the UI Servlet
- Checking Your Model in the Runtime Oracle Configurator

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 E-Business Suite 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 E-Business Suite Developer's Guide*. This document includes logging guidelines for both System Administrators and developers, and related topics.

- Oracle Application Framework Documentation Resources, Release 12, on MetaLink. This document provides troubleshooting information that is specific to the OA Framework.

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, page A-8.

- 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:

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

Checking the Response of the UI Servlet

What you are checking

Does the UI Servlet respond to a test message?

The test

Invoke the following URL in a Web browser:

Example
```
http://hostname:portnum/OA_HTML/configurator/UiServlet?test=version
```

where *hostname* is the name of your internet server and *portnum* is the port number for your Web listener. 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.

For example:

Example
```
http://www.mysite.com:8802/OA_HTML/configurator/UiServlet?test=version
```

If the UI servlet is running and installed correctly, entering this URL should produce a result similar to the following:

Example
```
Using configuration software build: 12.0.27.2
Expecting schema: 27b
```

> **Note:** The property `cz.uiservlet.versionfuncsavail` determines whether you can test the response of the servlet using the `test=veresion` string. For details, see cz.uiservlet.versionfuncsavail, page 4-14.

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 Application Framework Documentation Resources, Release 12, on MetaLink.

- There may be a basic problem with your Apache configuration. Refer to Oracle Application Server documentation for troubleshooting information.

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 Test Page for Invoking the JRAD Runtime Oracle Configurator, page 5-4, and the *Oracle Configurator Implementation Guide* for examples of simple test pages.

Test Page for Invoking the JRAD Runtime Oracle Configurator Example

```
<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>
```

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

   For details, see Verifying Apache and OC4J Setup, page 4-2.

2. 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 Test Page for Invoking the JRAD Runtime Oracle Configurator, page 5-4, click the button to display the specified User Interface.

A

Profile Options

This appendix describes the profile options that are used by Oracle Configurator and Oracle Configurator Developer.

This appendix covers the following topics:

- Overview of Profile Options

Overview of 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 Runtime Oracle Configurator Profile Options, page A-2. Only the System Administrator can view and update these profile options.

The profile options that are used by Oracle Configurator Developer are listed in Oracle Configurator Developer Profile Options, page A-5.

For information about setting profile options, see the *Oracle E-Business Suite User's Guide* .

Runtime Oracle Configurator Profile Options

Profile Option	Use r	Use r	Res p	Apps	Site	Required?	Default Value
BOM: Configurator URL of UI Manager , page A-8	-	X	X	X	X	Required with Order Management, *i* Store, TeleSales, and SalesOnLine	(none)
CZ: Add Model Routing Cookie, page A-9	-	0	0	0	X	Optional	False
CZ: Auto-Expire Discontinued IB Trackable Items, page A-9	-	X	X	X	X	Optional	Yes
CZ: Configurator Install Base, page A-10	-	X	X	X	X	Optional	oracle.apps. cz.dio.config.OracleIn stalledBase
CZ: Create Item Type Name Method, page A-10	-	0	0	0	X	Required with Oracle Bills of Material	Item Catalog Description
CZ: BOM Tree Expansion State, page A-10	-	X	0	0	X	Optional	One level
CZ: Disable Configurator Extensions, page A-11	-	0	0	0	X	Optional	No
CZ: Enable List Prices, page A-12	-	0	0	0	X	Optional	No

Profile Option	User	User	Resp	Apps	Site	Required?	Default Value
CZ: Enable ATP, page A-11	-	0	0	0	X	Optional	No
CZ: Enable Selling Prices, page A-7	-	0	0	0	X	Optional	No
CZ: BOM Tree Expansion State, page A-10	-	X	0	0	X	Optional	One level
CZ: Fail BV if Configuration Changed, page A-12	-	X	X	X	X	Required with Order Management	No
CZ: Fail BV if Input Quantities Not Maintained, page A-12	-	0	X	X	X	Required with Order Management	Yes
CZ: Generic Configurator UI Max Child Rows, page A-12	-	0	0	0	X	Required	50
CZ: Generic Configurator UI Type, page A-13	-	X	0	0	0	Optional	Java Applet
CZ: Include Unchanged Install Base Items, page A-13	-	0	0	0	X	Optional	Yes
CZ: Only Create CZ Config Items for Selected Nodes, page A-14	-	0	0	0	X	Optional	No

Profile Option	Use r	Use r	Res p	Apps	Site	Required?	Default Value
CZ: Populate Decimal Quantity Flags, page A-15	-	0	0	0	X	Optional	No
CZ: Publication Lookup Mode , page A-16	-	X	X	X	X	Optional	Production
CZ: Publication Usage , page A-17	-	X	X	X	X	Optional	Any Usage
CZ: Report All Baseline Conflicts, page A-17	-	X	X	X	X	Required with Oracle Install Base	No
CZ: Skip Validation Procedure, page A-18	-	0	0	0	X	Optional	(none)
CZ: Suppress Baseline Errors, page A-18	-	X	X	X	X	Required with Oracle Install Base	No
CZ: Use Alternate Retraction Algorithm Before Structure Changes, page A-19	-	0	0	0	X	Optional	No
CZ: Use Generic Configurator UI, page A-20	-	-	X	-	X	Required with *i* Store or Quoting	Yes

Note: Following is a description of the symbols used in the previous table:

X: You can update the profile option.

- (dash): You can view the profile value but you cannot change it.

Null/no value: You cannot view or change the profile option value.

The first User column refers to options that can be set indirectly through the Oracle Configurator Developer user responsibility. The other User column (and the Resp, App, and Site columns) refer to the System Administrator responsibility.

Oracle Configurator Developer Profile Options, page A-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*.

Oracle Configurator Developer Profile Options

Profile Option	User	User	Resp	Apps	Site	Required?	Default Value
CZ: BOM Node Display Name, page A-9	+	X	X	X	X	Required	Description
CZ: BOM Structure Display Method, page A-9	+	X	X	X	X	Required	Description
CZ: Custom Initialization Parameters , page A-10	+	X	X	X	X	Optional	(none)

Profile Option	User	User	Res p	Apps	Site	Required?	Default Value
CZ: Effectivity Date Filter, page A-11	+	X	X	X	X	Optional	All
CZ: Enable Creation of Functional Companions, page A-11	0	0	0	0	X	Optional	No
CZ: Non-BOM Node Display Name, page A-14	+	X	X	X	X	Required	Name
CZ: Non-BOM Structure Display Method, page A-14	+	X	X	X	X	Required	Name
CZ: Number of Table Rows Displayed, page A-14	+	X	X	X	X	Required	25
CZ: Number of Rows Displayed in Hierarchical Tables, page A-14	-	0	0	0	X	Optional	50
CZ: Require Locking, page A-17	-	0	0	0	X	Required	Yes
GMA: Default Language, page A-20	-	X	X	X	X	Required	US
Help System Root, page A-20	-	X	X	X	X	Required to display online help	CZ:CONTENTS (at Application level)

Note: Following is a description of the symbols used in the previous table:

X: You can update the profile option.

- (dash): You can view the profile value but you cannot change it.

+: A Configurator Developer user can view and change the value of this profile option.

Null/no value: You cannot view or change the profile option value.

The first User column refers to options that can be set indirectly through the Oracle Configurator Developer user responsibility. The other User column (and the Resp, App, and Site columns) refer to the System Administrator responsibility.

Profile Options

You must set a value for profile options followed by the word "required," no default is supplied. Ordinary users can see profile options followed by the word "exposed," only system administrators can see the rest. Further details follow the list, click an item to find them.

BOM: Configurator URL of UI Manager

CZ: Add Model Routing Cookie

CZ: Auto-Expire Discontinued IB Trackable Items

CZ: BOM Node Display Name

CZ: BOM Structure Display Method

CZ: BOM Tree Expansion State

CZ: Configurator Install Base

CZ: Create Item Type Name Method

CZ: Custom Initialization Parameters

CZ: Disable Configurator Extensions

CZ: Effectivity Date Filter

CZ: Enable ATP

CZ: Enable Creation of Functional Companions

CZ: Enable List Prices

CZ: Enable Selling Prices

CZ: Fail BV if Configuration Changed

CZ: Fail BV if Input Quantities Not Maintained

CZ: Generic Configurator UI Max Child Rows

CZ: Generic Configurator UI Type

CZ: Hide Focus in Generic Configurator UI

CZ: Include Unchanged Install Base Items

CZ: Non-BOM Node Display Name

CZ: Non-BOM Structure Display Method

CZ: Number of Rows Displayed in Hierarchical Tables

CZ: Number of Table Rows Displayed

CZ: Only Create CZ Config Items for Selected Nodes

CZ: Populate Decimal Quantity Flags

CZ: Publication Lookup Mode

CZ: Publication Usage

CZ: Report All Baseline Conflicts

CZ: Require Locking

CZ: Skip Validation Procedure

CZ: Suppress Baseline Errors

CZ: Use Alternate Retraction Algorithm Before Structure Changes

CZ: Use Generic Configurator UI

GMA: Default Language

Help System Root

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.

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:

Example

```
http://appsmachine.appsdomain:8800/OA_HTML/configurator/UiServlet
```

CZ: Add Model Routing Cookie

This profile option enables Oracle Configurator to add the cookie named `czPoolToken` to HTTP response messages. This cookie is used by the Web server at runtime to route specified Models to specified JVM pools.

The default value is False, which disables adding the cookie. Setting the value to True enables adding the cookie.

This profile can be set only at Site level.

For more information about routing Models to JVMs, see the *Oracle Configurator Implementation Guide*.

CZ: Auto-Expire Discontinued IB Trackable Items

This profile option controls whether an item's status automatically changes to Expired in Oracle Installed 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 Installed 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 Installed Base and Oracle Configurator, see the *Oracle Telecommunications Service Ordering Process Guide*. For more information about Oracle Installed Base, refer to Oracle Installed Base documentation.

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*.

CZ: BOM Structure Display Method

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

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

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*.

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 Installed Base. Oracle Installed 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 Installed Base, see the *Oracle Telecommunications Service Ordering Process Guide*.

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.

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*.

CZ: Disable Configurator Extensions

Set this profile option to `Yes` if you want to disable all Configurator Extensions in a runtime User Interface (and all Functional Companions in legacy UIs). Disabling all Configurator Extensions can be a useful way of debugging unexpected runtime behavior or performance issues.

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.

The default value is `No`, which means all Configurator Extensions are enabled.

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*.

CZ: Enable ATP

This profile option controls whether available to promise (ATP) information for selected items is displayed at runtime. To display ATP information in a UI, set this profile option to `Yes`. Also refer to the *Oracle Configurator Developer User's Guide*for details about settings that control whether ATP information appears in a generated UI.

For more information about integrating ATP with Oracle Configurator, see the *Oracle Configurator Implementation Guide.*

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

The default value is No.

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.

You would 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. However, as of Release 12.1, DHTML UIs are no longer supported. If you are using a User Interface generated with the HTML-based version of Oracle Configurator Developer, use Configurator Extensions, not Functional Companions. See Maintaining or Migrating Functional Companions, page 3-2.

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.

CZ: Enable List Prices

This profile option controls whether list prices for selected items are displayed at runtime. To display list prices in a UI, set this profile option to `Yes`. Also refer to the *Oracle Configurator Developer User's Guide*for details about settings that control whether list prices appear in a generated UI.

For more information about integrating pricing with Oracle Configurator, see the *Oracle Configurator Implementation Guide.*

The default value is No.

CZ: Enable Selling Prices

This profile option controls whether selling prices for selected items are displayed at runtime. To display selling prices in a UI, set this profile option to `Yes`. Also refer to the *Oracle Configurator Developer User's Guide* for details about settings that control whether selling prices appear in a generated UI.

For more information about integrating pricing with Oracle Configurator, see the *Oracle Configurator Implementation Guide.*

The default value is No.

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.

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.

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*.

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.

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*.

CZ: Include Unchanged Install Base Items

When an Oracle Installed 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 Installed 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 Installed Base and Oracle Configurator, see the *Oracle Telecommunications Service Ordering Process Guide*.

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*.

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*.

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.

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*.

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.

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.

CZ: Populate Decimal Quantity Flags Profile Option, page A-16 describes the effect of setting this profile option.

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 Oracle Configurator Servlet Considerations , page 4-1.

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.

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 User, Responsibility, Application, and Site levels. The default value at the Site level is `Production.`

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

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*.

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 Installed 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 CZ: Suppress Baseline Errors, page A-18.

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

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*.

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:

Example

```
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*.

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 Installed 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 Installed Base.)

When an end user reconfigures an instance of a configuration, Oracle Configurator compares it to the baseline configuration in Oracle Installed 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 Installed 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 CZ: Report All Baseline Conflicts, page A-17.

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

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.

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 CZ: Generic Configurator UI Type, page A-13.

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

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 Configuring Oracle Configurator Developer for Multiple Language Support, page 2-5.

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.

Glossary

This glossary contains definitions relevant to working with Oracle Configurator.

A

Archive Path

The ordered sequence of Configurator Extension Archives for a Model that determines which Java classes are loaded for Configurator Extensions and in what order.

B

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.

batch validation

A background process for validating selections in a configuration.

binding

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

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 (Assemble To Order) 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 options: true or false.

C

CDL (Constraint Definition Language)

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

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

CIO (Oracle Configuration Interface Object)

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.

command event

An event that is defined by a character string and detected by a command listener.

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 Features, Totals, Resources, 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 Features in the Model to control the allowable combinations of Options. *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 Options 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 Features, that is configurable and instantiable. An Oracle Configurator Developer node type that represents a configurable element of a Model.

Component Set

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

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 model

Represents all possible configurations of the available options, and consists of model structure and rules. It also commonly includes User Interface definitions and Configurator Extensions. 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 configurations. *See also* rules.

configuration session

The time from launching or invoking to exiting Oracle Configurator, during which end users make selections to configure an orderable product. A configuration session is limited to one configuration model that is loaded when the session is initialized.

configurator

The part of an application that provides custom configuration capabilities. Commonly, a window that can be launched from a host application so end users can make selections resulting in valid configurations. *Compare* Oracle Configurator.

Configurator Developer

See OCD.

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 classes that implement Configurator Extensions.

connectivity

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

See CDL

Container Model

A type of BOM Model that you import from Oracle Bills of Material into Oracle Configurator Developer to create configuration models that support connectivity and contain 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 options. *Compare* instance.

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.

D

default

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 Features or Options 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).

Design Chart

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

E

element

Any entity within a model, such as Options, Totals, Resources, UI controls, and components.

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.

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 Features or Options 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.

F

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 Options.

G

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 rules that automatically select some product options and exclude others based on the end user's responses.

H

host application

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

I

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 Features or Options 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, Migrate, and Synchronization concurrent processes. The import server is sometimes referred to as the remote server.

initialization message

The XML (Extensible Markup Language) 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

A runtime occurrence of a component in a configuration that is determined by the component node's Instance attribute specifying a minimum and maximum value. *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.

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.

L

listener

A class in the CIO that detects the occurrence of specified events in a configuration session.

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 Features and Options 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.

M

model

A generic term for data representing products. A model contains elements that correspond to items. Elements may be components 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

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

model structure

Hierarchical "tree" view of data composed of elements (Models, Components, Features, Options, BOM Models, BOM Option Class nodes, BOM Standard Item nodes, Resources, and Totals). May include reusable components (References).

N

Negates relation

A type of Oracle Configurator Developer Logic Rule type that determines the logic state of Features or Options 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). *Compare* 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.

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.

O

object

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

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 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 Developer

The tool in the Oracle Configurator product used for constructing and maintaining configuration models.

Oracle Configurator engine

The part of the Oracle Configurator product that uses configuration rules to validate runtime selections. Compare generated logic. *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 users to make the selections of a configuration.

P

Populator

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

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 Options are specified implicitly by relationships among Property values of the Options.

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.

R

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 Models and shared objects in a single location.

Requires relation

An Oracle Configurator Developer Logic Rule relationship that determines the logic state of Features or Options 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

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 Rules to contribute to and consume from a Resource.

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

rules

Also called business rules or configuration rules. 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 Components, Features, and Options. 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 in which an implementer (tester), end user, or customer configures a product whose model was developed in Oracle Configurator Developer. *See also* configuration session.

S

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.

T

termination message

The XML (Extensible Markup Language) 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.

U

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.

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 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 Interface

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

V

validation

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

W

Workbench

Set of pages in Oracle Configurator Developer for creating, editing, and working with Repository objects such as Models and UI Templates.

Index