Oracle® Configurator

Custom Web Deployment Guide

Release 11*i*

June 2001

Part No. A87530-02

This document describes how to embed the Oracle Configurator (OC) into an Internet host application, such as a custom web store, using Oracle Configurator integration technology.

ORACLE®

Oracle Configurator Custom Web Deployment Guide, Release 11*i*

Part No. A87530-02

Primary Author: Mark Sawtelle

Contributing Authors: Tina Brand

Contributors: Brent Benson, Mike Colena, David Gosselin, CK Jeyaprakash, Manoj Jose, David Kulik, David Lee, Keri-Lyn Mullis, Janet Page, Marty Plotkin, Mike Sheehy, Sean Tierney, Lois Wortley

Contents

3 Session Initialization

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle Configurator Custom Web Deployment Guide, Release 11*i*

Part No. A87530-02

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

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

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments through your call to Oracle Support Services or by sending them to:

Oracle Configurator
Oracle Corporation
Documentation
21 North Avenue
Burlington, MA 01803
USA

If you would like a reply, please give your name, address, and telephone number below.

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

Preface

Oracle Configurator (OC) is an Oracle product that enables end users of a web-based application to configure products and services.

This manual describes how to embed the run-time Oracle Configurator into an Internet application, such as a custom web store.

You can see the run-time Oracle Configurator in use in a number of Oracle's Internet applications, such as Oracle iStore and Sales Online.

Intended Audience

Technologies and Tools

Working with the OC requires varying degrees of familiarity with the following technologies and tools:

- web browsers
- Internet application servers (web servers)
- servlets
- web stores
- Java
- JavaScript
- DHTML
- HTML

- XML and XSL

Related Documents

For more information, see the following manuals in the Oracle Configurator documentation set:

- *Oracle Configurator Developer User's Guide*
- *Oracle Configurator Implementation Guide*
- *Oracle Configurator Installation Guide*
- *Oracle Configuration Interface Object (CIO) Developer's Guide*

The following documents may also be useful:

- *Oracle8i JDBC Developer's Guide and Reference*

Structure

This manual contains the chapters and appendices listed in the following table:

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 following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
...	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
italics	Italic type in text or tables 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.

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) via Metalink, Oracle's customer support Web site, at:

`http://metalink.oracle.com`

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

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

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

For general information about Oracle Applications, visit AppsNet at:

`http://www.oracle.com/appsnet/index.html`

Accessibility

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our

documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program web site at:

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

Accessibility of Code Examples in Documentation

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

1

Introduction

1.1 Typical Tasks

This document will help you complete the tasks that you must perform in order to integrate the Oracle Configurator (OC) into your internet host application.These tasks are listed in the following table:

	For this task ...	See ...
Develop the Configuration Window	In Oracle Configurator Developer, construct your Model and Rules.	*Oracle Configurator Developer User's Guide*
	In Oracle Configurator Developer, create and modify your User Interface.	
	Test Model, Rules, and User Interface in Oracle Configurator Developer	

	For this task ...	See ...
Customize and Deploy the web-based Configuration Window	Install the Oracle Configurator servlet on your internet application server.	Chapter 2, "Servlet File Information"
	Tailor the initialization message that invokes the configuration window and the return URL that handles the configuration outputs.	Chapter 3, "Session Initialization"
	Extract the required data from the XML termination message produced by the configuration window.	Chapter 4, "Session Termination"
	Validate configurations without direct user interaction.	Chapter 5, "Batch Validation"
	Customize certain HTML files to modify the appearance of the configuration window.	Chapter 6, "User Interface"
	Provide pricing and ATP data in the configuration window.	Chapter 7, "Pricing and ATP"
	Test the configuration window in the host application, running in an internet browser.	*Oracle Configurator Developer User's Guide*

1.2 Oracle Configurator Architecture

1.2.1 Three-tier Web Application Architectures

Oracle Configurator allows you to deploy a web-based configuration window in a three-tier architecture. Figure 1–1 describes these tiers.

Table 1–1 Three-tier web application architecture

Tier	Tier components	OC Components
Client	thin client	your host application, running in a browser OC configuration window (DHTML)

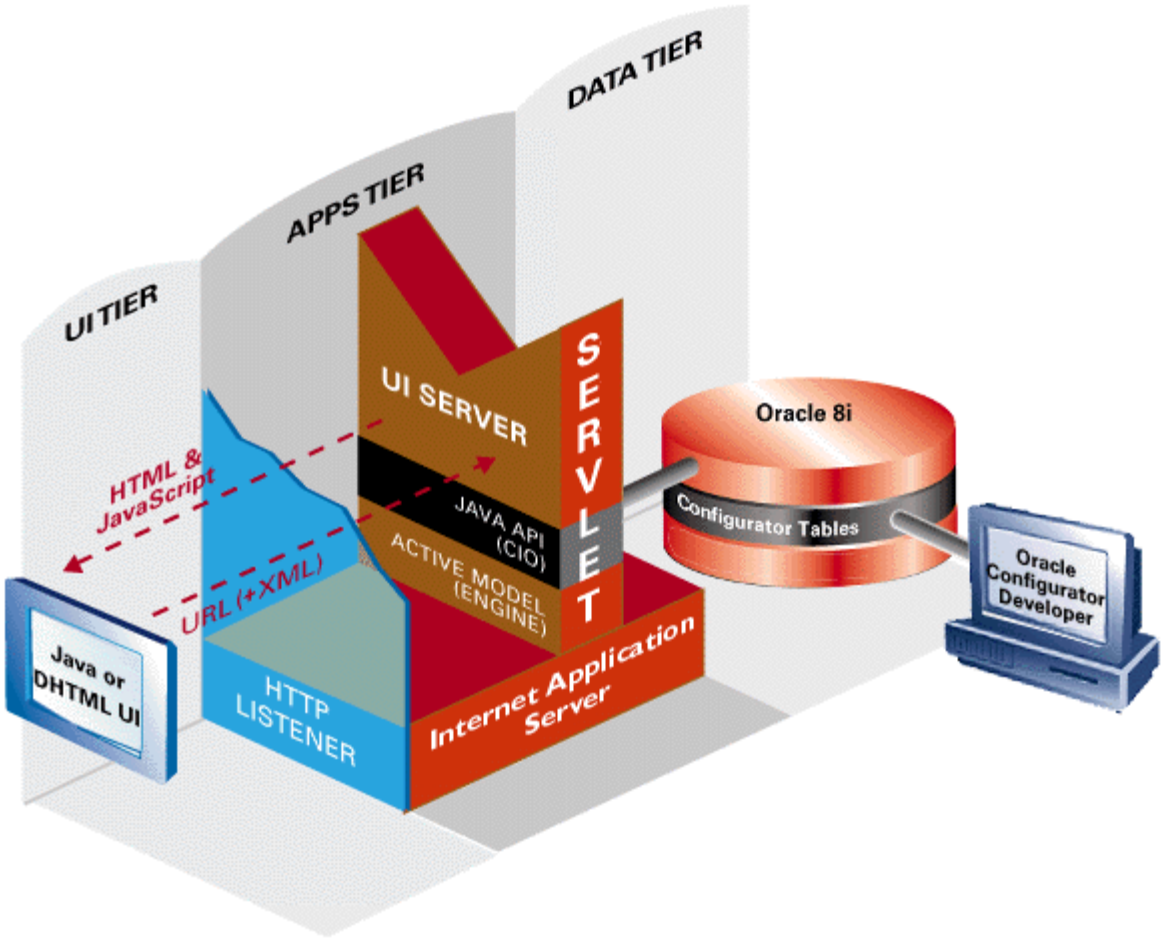
Table 1–1 Three-tier web application architecture

Tier	Tier components	OC Components
Application	application (web) server	Oracle Apache
	application logic	Oracle Configurator Active Model
Database	application database	Oracle8*i* Enterprise Edition, Oracle Configurator schema

In the client tier, a host application running in a browser displays the configuration window, which in turn communicates with the Oracle Configurator engine and the Oracle Configurator schema.

An application server brokers the HTML, XML, and data queries that provide the configuration window with content from the Oracle Configurator schema. A web server brokers the connection (HTTP). Figure 1–1 on page 1-4 illustrates this architecture.

Figure 1–1 Architectural Overview of Oracle Configurator



The OC Servlet is installed on the application server, along with the other server-side components of Oracle Configurator. The database should not be on the same machine.

The application server is the critical component that enables the thin-client browser to maintain a web interface to the Configuration Model accessed by the configuration window. The servlet running on the application server interfaces with

the UI Server. The UI Server generates a JavaScript rendering of the Active Model and Active UI.

1.2.2 Application Flow

1. An initialization message is sent to the OC Servlet. The initialization includes parameters that specify a database connection (such as user, password, database) and parameters that identify the configuration model required to drive the configuration window (such as an ID for the model, or a particular saved configuration).
2. If a configurator session is not already underway for this user, the servlet creates a new UI session.

   On initial load, all mandatory components are loaded into memory and remain in memory for the entire session. Optional configurable components are loaded on demand and will be unloaded from memory as required.
3. The OC Servlet holds the session information in its session object for that user, and passes all XML messages to the UI Server.
4. The UI Server produces JavaScript.
5. JavaScript is substituted into the HTML template file as the response to the user's request, and displayed in the client browser.
6. Messages from the configuration window (client) to the UI Server are generic input events (click or data input) on a particular control ID within a particular session, stored in the initialization message. Results of a processed event are rendered as JavaScript, to describe changes to the client configuration window such as navigation results, altered logic states, changed counts, etc.

1.3 Elements Needed and Provided

Table 1–2 lists all the elements needed by a web application using Oracle Configurator, with a brief description of how you use each in building your host application that incorporates the Oracle Configurator configuration window.

Table 1–2 Elements and Actions Needed and Provided in Oracle Configurator

Element Needed	Provided by Oracle	Provided by You	Action
Your host application (e.g., Oracle iStore or other web store)	No	Yes	You implement the application.
Web server (e.g., Apache)	No	Yes	You install and maintain the server.
Oracle Applications database (including Oracle Configurator schema tables)	No	Yes	You provide connectivity between the database and the OC Servlet.
Oracle Configurator OC Servlet	Yes	No	The OC Servlet encapsulates other elements: ▪ the UI Server ▪ the CIO ▪ the logic engine
Oracle Configurator UI Server	Yes	No	Automatically accessed by OC Servlet to process user interface
Oracle Configurator CIO	Yes	No	Automatically accessed by OC Servlet. (Also available for custom programming.)
Oracle Configurator logic engine	Yes	No	Automatically accessed by CIO, through OC Servlet.
DHTML configuration window	Yes	No	Configuration window is produced automatically by OC Servlet.
Configuration window UI layout	Yes	No	Layout of UI elements is automatically determined by Model, stored in Oracle Configurator schema.
Default HTML templates for configuration window	Yes	No	Configuration window automatically displays UI in default format.
Customized HTML templates for configuration window	No	Yes	You modify the customizable elements of HTML templates to modify appearance of configuration window, or create new templates following documented structure.

Table 1–2 (Cont.) Elements and Actions Needed and Provided in Oracle Configurator

Element Needed	Provided by Oracle	Provided by You	Action
Image files, for the background and UI controls of the configuration window.	Yes	No	Image files appear in HTML templates and JavaScript controls. (You can customize the set of files.)
XML initialization message for the OC Servlet	No	Yes	You specify the XML parameters in the initialization message that your application passes to the configuration window.
XML termination message from the OC Servlet	No	Yes	You parse the XML output from the UI Server's termination message, and pass data back to your host application.
"Return URL" Java servlet	No	Yes	You implement a servlet class that provides behavior after the configuration window is used.
Integration code for order entry API	No	Yes	You implement the integration code.

2

Servlet File Information

In order to use the run-time Oracle Configurator, you must first install the OC Servlet on your Internet server. This process is described in the *Oracle Configurator Installation Guide*. Certain post-installation tasks are described in the *Oracle Configurator Implementation Guide*.

In order to customize Oracle Configurator in your host application, you may need to modify certain Oracle Configurator files. This chapter describes the location and purpose of those files.

2.1 Related Documentation

You will need to consult the following documentation when installing the run-time Oracle Configurator servlet:

- *Oracle Configurator Installation Guide*
- *Oracle Configurator ReadMe*
- *Oracle Configurator Release Notes*
- *Oracle Configurator Implementation Guide*

2.2 Prerequisites

- You must have installed Oracle Applications Release 11*i*, specifically including Oracle Configurator.
- You must have installed your internet server. Oracle Configurator has been tested with Oracle Internet Application Server (iAS).
- You must have JDBC connectivity between your internet server and your Oracle database.

- Check the *Oracle Configurator Release Notes*, and the chapter on deployment in the *Oracle Configurator Implementation Guide*, for any other requirements.

2.3 Installing the Required Files

The required files must be in place before installing the servlet in your Internet server.

- To install the required files on a Solaris machine, run Oracle Rapid Install and perform the subsequent steps, as described in the *Oracle Configurator Installation Guide*.
- You may wish to move certain files to other locations, to suit the needs of your site or host application. If so, see Section 2.4, "Required Files and Locations" on page 2-2.

2.4 Required Files and Locations

There may be reasons for moving certain installed files to alternate locations.

- To suit the needs or working conventions of your site.
- To suit the needs of your host application.

In order to assist you in moving these files correctly, this section describes constraints and guidelines on their location.

The files are also enumerated, in order to assist you in troubleshooting simple problems arising from missing or misplaced files.

2.4.1 General Directory Structure

Table 2–1 shows the directories required for the run-time Oracle Configurator, and their relationship. This general structure applies to all platforms, though the details may vary by platform. In some cases, the same physical directory may fill more than one role.

Table 2–1 General Structure of Directories for Oracle Configurator

Directory Role	Description
OC Installation	This directory is referenced in these instructions as *oc_install*. The directory in which you install OC, based on your choice of installation directory in the Oracle Configurator setup program. On Solaris, this defaults to $APPL_TOP.
Servlet	Contains the Java class or archive files that implement the OC Servlet.
HTML	Contains the HTML template files that define the run-time Oracle Configurator of your host application (see Chapter 6).
Media	Contains the image files used by the run-time Oracle Configurator of your host application.
Log	Contains log files written by the OC Servlet when the run-time Oracle Configurator is used.

Note that it is not strictly necessary for the Servlet directory to have a separate physical location, since the files it contains are referenced by environment variables that you set while installing the run-time Oracle Configurator servlet.

Directory and file names are case-sensitive on Solaris.

Certain HTML files are language-specific, and are stored in language-specific directories, such as OA_HTML/US. See Table 2–4 on page 2-5 for a list of these files.

2.4.2 File Types

Table 2–2 shows the types of files included in an installation of the run-time Oracle Configurator.

Table 2–2 File Types

Type	Filename Extension	Description
HTML Template	.htm, .html	HTML files that provide the layout and interface for the configuration window. Certain of these can be customized.
Java Archive	.jar, .zip	Java archive files that implement the OC Servlet, UI Server, configuration engine, and other elements of the run-time Oracle Configurator.
Java Class	.class	Java class files that implement application logic.

Table 2–2 (Cont.) File Types

Type	Filename Extension	Description
Java Source	.java	Java source code files that can be modified to customize the behavior of the OC Servlet with your host application.
JavaScript Control	.js	JavaScript control files that implement the behavior of the run-time Oracle Configurator.
Media	.gif, .jpg	Image files used by your host application.
Shared Object	.so, .dll	The platform-specific implementation of the servlet.

2.4.3 Files for the Servlet Directory

Table 2–3 shows the files that should be installed in your Servlet directory.

The Servlet directory contains files that must be referenced in the PATH and CLASSPATH environment variables. It is not strictly necessary for the Servlet directory to have a distinct physical location. For instance, its files can be located directly in the *oc_install* directory, rather than in a separate `/servlets` subdirectory.

Table 2–3 Files for the Servlet Directory

File	For Platform	Comment
Java Classes		
apps.zip	All	Includes all the Java classes required for OC, in addition to those for Oracle Applications.
xmlparserv2.zip	All	Includes the Java classes for the XML parser.
jdbc111.zip	All	Includes the Java classes for JDBC database connectivity.
Shared Object		
cz.dll	Windows NT	Must be in the PATH system environment variable on the host machine on which the servlet is installed. This should be set by the OC installation program.
czjni.dll	Windows NT	
libcz.so	Solaris	Must be in the LD_LIBRARY_PATH environment variable parameter for your servlet.
libczjni.so	Solaris	

2.4.4 Files for the HTML Directory

The files listed in Table 2–4 must be located in the HTML directory, on all platforms.

Table 2–4 Files for the HTML Directory

File
HTML Templates
czCntnt.htm
czdisp.htm
czHeartBeat.htm
czSource.htm
cztree.htm
Language-Specific (NLS) HTML Templates
czFraIE.htm
czFraNS.htm
czHeader.htm
czLeft.htm
czRight.htm
czButtonBar.htm
czFraTemplate.htm
czIFrame.htm
JavaScript Controls
czAll.js

If you are implementing Oracle Configurator in a language other than English, you may need to translate the NLS files.

2.4.5 Files for the Media Directory

The image files in the Media directory are used by the run-time Oracle Configurator embedded in your host application to decorate the DHTML configuration window, and also to represent application logic state in the DHTML controls in the configuration window.

These files must be compatible with web browser technology. You cannot use BMP (Windows bitmap) files in your user interface for the configuration window, since this file format is not compatible with web browsers.

This will affect you if the User Interface that you defined in Oracle Configurator Developer included any BMP files.

The configuration window can use GIF, JPG, and other formats compatible with web browsers.

3

Session Initialization

3.1 Overview of this Chapter

3.2 How it Works

In order to prepare a Model so that it can be configured in the configuration window, you must first either:

- Import BOM and item data into the Oracle Configurator schema so that it is available to Oracle Configurator Developer. Importing is described in the *Oracle Configurator Implementation Guide*.
- Use Oracle Configurator Developer to develop a Model and its associated Configuration Rules and a User Interface.

In order to integrate the configuration window into your host application, the application must:

- Use frames, one of which contains the configuration window.
- Provide a means of posting the initialization message.
- Provide a return URL servlet to process results of your user's selections in the configuration window.

In a typical host application (such as a web store), a button, tab, or similar control is coded so that it causes the configuration window to appear, and to contain the appropriate user interface. For the purposes of this explanation, think of this control as "the Configure button". You intend for your user to click it at a point where configuration of a product is required.

Oracle Configurator provides you with:

- A **servlet**, the Oracle Configurator Servlet, that handles interaction between the host application, the configuration window, and the product Model that you define in Oracle Configurator Developer (which contains product structure, logic rules, and user interface definitions).
- The host application invokes the URL of the OC Servlet with a query string that contains an XML **initialization message**. It does this by setting the location property of the frame where the Configurator is to be displayed. You tailor this initialization message to specify a number of important parameters that govern the behavior of the configuration window. These parameters are described in this chapter.
- A set of **HTML Template files** that interact with the OC Servlet. You can customize these files to suit them to the needs of your host application, or leave them as they are. See Chapter 6, "User Interface" for details.

See Section 1.2, "Oracle Configurator Architecture" on page 1-2 in Chapter 1, "Introduction" for an explanation of the interaction between all these elements.

3.3 What You Do

Integrating the configuration window with your host application consists primarily of causing your host application (e.g., through the coding of the "Configure" button) to post the XML initialization message to the OC Servlet.

You must first install the OC Servlet, as described in the *Oracle Configurator Installation Guide*. There is additional information on the files that are installed, in Chapter 2, "Servlet File Information" of this document.

There are two basic situations that the host application must handle when integrating the configuration window, as shown in the following table:

You need to handle this...	As described in...
Initialization of the configuration window, to prepare it for your user to perform configuration selections.	Section 3.4, "Definition of Session Initialization" on page 3-3
Termination of the configuration window, to return control and results to the host application when your user closes the window.	Section 4.3, "Definition of Session Termination" on page 4-1

3.3.1 Responsibilities of the Host Application

The responsibilities of the host application while the configuration window is in use are:

- Block the host application while the configuration window is in use, disallowing all user input and navigation to or from the configuration window.
- Disable visible functions in the surrounding host application that would confuse the user while interacting with the configuration window.
- Handle the output from the **return URL**, and close the configurator window by resetting its frame's location property.

3.4 Definition of Session Initialization

Session initialization takes place when your host application invokes the OC Servlet which opens the configuration window.

When you set the parameters of the initialization message in your host application, your parameters handle the types of responsibilities listed in Section 3.6, "Initialization Parameter Types" on page 3-8.

When your host application invokes the configuration window, the initialization message is sent to the OC Servlet, using the HTTP POST method. (POST is used in preference to GET to accommodate the length of the message.)

The initialization message is written in XML, and has `<initialize>` as its document element. You must specify the parameters for `<initialize>` in order to determine the state in which the configuration window will open.

3.5 Setting Parameters

You specify `<initialize>` and its parameters as the value of an XML message that is passed to the OC Servlet. The OC Servlet is invoked through its URL, which is determined when you install it (as described in the *Oracle Configurator Installation Guide*). By default, the URL is:

hostname:*port*/*virtual_path_of_servlet_class*/oracle.apps.cz.servlet.UiServlet

For example:

```
http://www.mysite.com:8802/configurator/oracle.apps.cz.servlet.UiServlet
```

For use with Oracle Configurator, the *virtual_path_of_servlet_class* is always set to `configurator`.

3.5.1 Parameter Syntax

All parameters to the XML initialization message are specified as name-value pairs, using attributes of the `<param>` document element, in the form:

```
<param name="parameter_name">parameter_value</param>
```

Example 3–1 shows the basic syntax for specifying the OC Servlet's URL and the initialization message as you would typically use them in your host application. The parts that you need to modify are typographically emphasized.

Example 3–1 Syntax of initialization message in HTML context

```
...
<form action="servlet_URL" method="post" >
<input type=hidden name="XMLmsg" value=
'<initialize>
<param name="parameter_1_name">parameter_1_value</param>
<param name="parameter_n_name">parameter_n_value</param>
</initialize>'>
```

```
<input type="submit" value="Configure">
</form>
...
```

When your user clicks the "Configure" button produced by the second `INPUT` element in Example 3–1, the initialization message is posted to the OC Servlet.

See Example 3–2 for some typical values for the parameters, and Example 3–3 for a test page that puts the values in context.

Be aware that XML permits you to use either single or double quotation marks around the value of an element's attribute, so you might also write:

```
"<initialize>
    <param name='parameter_name'>parameter_value</param>
</initialize>"
```

3.5.1.1 Omitting Parameters and Values

If you omit a parameter entirely from the initialization message, then the parameter is ignored by the OC Servlet.

However, if a parameter has a default value, then you must either accept the effect of the default, or override the default with a specified value. The default values for the parameters are provided in Section 3.7, "Initialization Parameter Descriptions" on page 3-15.

> **Note:** Do not leave the value of a parameter empty. Doing so will cause an error when the initialization message is processed.

If you omit the value of a parameter, then the OC Servlet will generate an error message indicating which parameter is missing a value. The message appears in the browser window, and in the servlet's session log.

3.5.2 Typical Parameter Values

Example 3–2 shows an example of a basic set of initialization parameters that cover all of the types of responsibility shown in Table 3–2 on page 3-8.

See Section 3.7, "Initialization Parameter Descriptions" on page 3-15 for the complete list of valid parameters to the initialization message.

See Example 3–1 for the syntax of the initialization message, and Example 3–3 for a test page that puts the values in context.

Example 3–2 Basic XML initialization parameters

```
<initialize>
  <param name="two_task">vis11</param>
  <param name="gwyuid">applsyspub/pub</param>
  <param name="fndnam">apps</param>
  <param name="user">mfg</param>
  <param name="pwd">welcome</param>
  <param name="ui_type">DHTML</param>
  <param name="ui_def_id">3120</param>
  <param name="return_
url">http://www.mysite.com:10130/configurator/Checkout</param>
</initialize>
```

Table 3–1 explains the parameters used in Example 3–2.

Table 3–1 Explanation of initialization parameters in Example 3–2

Parameter type	Name	Description
Login	two_task	The name of the database instance to connect to. This value can be read from the environment variable TWO_ TASK.
Login	gwyuid	The gateway user ID required for authentication with the Oracle Applications protocol.
Login	fndnam	Used by Oracle Configurator to authenticate standard Oracle Applications users through the FND login algorithms.
Login	user	The user ID of the login user.
Login	pwd	The password of the login user.
Login	ui_type	The type of web pages to be returned by the OC Servlet.
Configuration Identification	ui_def_id	The UI Definition ID of the User Interface that you created in Oracle Configurator Developer. There are several ways to specify a Configuration choice. See Section 3.6.3, "Model Identification Parameters" on page 3-9.
Return	return_url	The URL of the Java servlet that processes the configurator's termination message.

3.5.3 Minimal Test of Initialization

Example 3–3 shows the HTML for a minimal web page that invokes the configuration window. You can use this test page as a stand-in for your host application.

This example includes the invocation of the OC Servlet as shown in Example 3–1 and the initialization message parameters as shown in Example 3–2.

Example 3–3 Minimal HTML for invoking the configuration window

```
<html>
<head>
<title>Minimal Configurator Test</title>
</head>
<body>
<form
action="http://www.mysite.com:10130/configurator/oracle.apps.cz.servlet.UiServle
t" method="post">
<input type="hidden" name="XMLmsg" value=
'<initialize>
<param name="two_task">vis</param>
<param name="gwyuid">applsyspub/pub</param>
<param name="fndnam">apps</param>
<param name="user">mfg</param>
<param name="pwd">welcome</param>
<param name="ui_type">DHTML</param>
<param name="ui_def_id">3120</param>
<param name="return_
url">http://www.mysite.com:10130/configurator/Checkout</param>
</initialize>'>
<p>Click button to configure model...
<input type="submit" value="Configure">
</form>
</body>
</html>
```

3.5.4 Parameter Validation

When your host application invokes the OC Servlet, the UI Server validates the parameters of the initialization message.

- There must be a way of connecting to the database, such as the parameters `two_task` or `alt_database_name`.

- There must be a way to choose a Model to be configured, so there must be one of the combinations described in Section 3.6.3, "Model Identification Parameters" on page 3-9.

If there is a problem processing the initialization message, then the UI Server returns a termination object to the OC Servlet, which returns it to the host application or displays the results to your user, through the URL specified in the `return_url` parameter.

3.6 Initialization Parameter Types

This section describes the use of the types of initialization parameters listed in Table 3–2 on page 3-8. All of the initialization parameters are described alphabetically in Section 3.7, "Initialization Parameter Descriptions" on page 3-15.

Table 3–2 Types of Initialization Parameters

Type	Required?	Description	See
Login	Yes	Information required for access to the proper data, such as database, user, and password.	Section 3.6.1, "Connection Parameters" on page 3-9
Configuration	Yes	Identification of the Model to be configured, or of the existing configuration to be modified. See Section 3.6.3, "Model Identification Parameters" on page 3-9.	Section 3.6.3, "Model Identification Parameters" on page 3-9
Return	No, but recommended	Identification of the return URL that handles the results from the configuration window, such as configuration outputs.	Section 3.6.4, "Return URL Parameter" on page 3-11
Pricing and ATP	No	Identification of the procedures and interfaces to be used for obtaining prices and ATP dates.	Section 3.6.5, "Pricing Parameters" on page 3-12 Section 3.6.6, "ATP Parameters" on page 3-13
Other	No	Miscellaneous desired information.	Section 3.6.7, "Arbitrary Parameters" on page 3-14

3.6.1 Connection Parameters

In order to connect the configuration window to the database, you must specify one of the following parameters or combinations of parameters in your initialization message.

- database_id
- database_id and icx_session_ticket
- alt_database_name, user, and pwd
- two_task, user, pwd, gwyuid, and fndnam

For descriptions of the individual parameters, see Section 3.7, "Initialization Parameter Descriptions" on page 3-15.

3.6.2 Applicability and Publishing Parameters

In order to determine the Model to use if it has been published, you must specify one or more of the following parameters or combinations of parameters in your initialization message:

- calling_application_id
- config_model_lookup_date
- config_effective_usage
- publication_mode

3.6.3 Model Identification Parameters

There are several different ways in which you can identify the Model to be configured, or the existing configuration to be modified. In your initialization message, you must use one of the parameters or combinations of parameters listed in Table 3–3:

Table 3–3 Configuration Identification Parameters

Method for Configuration Identification	Initialization Parameters
Section 3.6.3.1, "Identifying the User Interface Definition"	ui_def_id

Table 3–3 (Cont.) Configuration Identification Parameters

Method for Configuration Identification	Initialization Parameters
Section 3.6.3.2, "Identifying the Configuration"	config_header_id config_rev_nbr
Section 3.6.3.3, "Identifying the Model"	organization_id inventory_item_id
(These parameters are from previous releases, retained for backward compatibility.)	context_org_id model_id
Section 3.6.3.4, "Identifying the Product"	product_id

For descriptions of the individual parameters, see Section 3.7, "Initialization Parameter Descriptions" on page 3-15.

3.6.3.1 Identifying the User Interface Definition

Parameter to specify:

- ui_def_id

Using this parameter will result in creating a new configuration. It is most useful for identifying a Model created entirely in Oracle Configurator Developer. It is also useful for specifying a particular UI out of several that may be available for a Model, whether or not the Model was created entirely in Configurator Developer.

This ID identifies a User Interface created in Configurator Developer. The User Interface includes identification of the Model to be configured (which is associated with Configuration Rules and the Project in Configurator Developer in which they were developed).

3.6.3.2 Identifying the Configuration

Parameters to specify:

- config_header_id
- config_rev_nbr

Using this combination of parameters will result in restoring an existing configuration.

The configuration header ID is the main identifier of an existing configuration record previously created and saved by your host application or another application

that knows how to save configurations to the Oracle Configurator schema, such as the run-time Oracle Configurator. The configuration revision number distinguishes among particular saved configurations sharing the same header information.

3.6.3.3 Identifying the Model

For backward compatibility with previous releases, you can specify these parameters:

- context_org_id
- model_id

For the current release, you should specify these parameters:

- organization_id
- inventory_item_id

Using this combination of parameters will result in creating a new configuration. It is only useful for identifying a Model that was originally created in another application (such as Oracle Applications Bills of Materials) and then imported into Oracle Configurator Developer.

Your host application must determine which Model to configure and be able to identify it by inventory_item_id and organization_id.

3.6.3.4 Identifying the Product

- product_id

If you created a Model in Oracle Configurator Developer, and entered a Product ID when you published the Model, then specify the product_id in your initialization message.

3.6.4 Return URL Parameter

The return URL is the fully qualified URL of a Java servlet installed on your web server that implements the behavior that you want after the user has ended the configuration session.

- return_url

```
<param name="return_
url">http://www.mysite.com:10130/configurator/Checkout</param>
```

The example parameter above comes from Example 3–3, "Minimal HTML for invoking the configuration window" on page 3-7.

The URL specification in the `return_url` parameter must stop at the name of the servlet class. You cannot pass parameters to the class in this URL (for instance, with the *`classname`*?*`parameter`*=*`value`* syntax). The return URL servlet should only get data from the termination message, which is passed to the it as the value of the XMLmsg argument.

The initialization message is sent to the return URL. This occurs in the event of normal termination, cancellation by the end user, or exceptions.

The return servlet is installed in your web server's Servlet directory, as described in Section 2.4.3, "Files for the Servlet Directory" on page 2-4 in Chapter 2, "Servlet File Information".

See Section 4.8, "The Return URL" on page 4-9 for details on the implementation of the return servlet.

3.6.5 Pricing Parameters

See Chapter 7, "Pricing and ATP" for details on the use of these parameters. See Appendix B, "Examples of Pricing and ATP Callback Procedures" on page B-1 for examples.

Choose these pricing parameters, depending on which pricing methods are required for your host application, as indicated in the following table:

Pricing method	Use these parameters...
Advanced Pricing (QP), or your own callback pricing procedures that call it	pricing_package_name and the associated parameters listed under Section 3.6.5.1, "Oracle Applications Release 11i Pricing Parameters" on page 3-12

For descriptions of the individual parameters, see Section 3.7, "Initialization Parameter Descriptions" on page 3-15.

3.6.5.1 Oracle Applications Release 11*i* Pricing Parameters

Since these parameters are designed to be used with an interface using callback procedures, they are also referred to as "callback pricing" parameters.

To use callback pricing, provide these parameters:

- pricing_package_name
- configurator_session_key
- either price_mult_items_proc, price_mult_items_mls_proc, or price_single_item_proc

These parameters are used when the configuration window calls existing APIs to get pricing and ATP data for configured items.

These parameters are accessible with the Configuration Interface Object (CIO) method `Configuration.setInitParameters()`, described in the API reference portion of the *Oracle Configuration Interface Object (CIO) Developer's Guide*.

3.6.6 ATP Parameters

See Chapter 7, "Pricing and ATP" for details on the use of these parameters. See Appendix B, "Examples of Pricing and ATP Callback Procedures" on page B-1 for examples.

Choose from ATP (Available To Promise) parameters, depending on which ATP methods are required for your host application, as indicated in the following table:

ATP method	Use these parameters...
"callback" interface	atp_package_name and the associated parameters listed under Section 3.6.6.1, "Oracle Applications Release 11i ATP Parameters" on page 3-13

For descriptions of the individual parameters, see Section 3.7, "Initialization Parameter Descriptions" on page 3-15.

3.6.6.1 Oracle Applications Release 11*i* ATP Parameters

Since these parameters are designed to be used with an interface using callback procedures, they are also referred to as "callback ATP" parameters.

To use callback ATP, provide these parameters:

- atp_package_name
- configurator_session_key
- get_atp_dates_proc
- requested_date (optional, defaults to SYSDATE)

- warehouse_id
- and one of the following:
  - customer_id and customer_site_id
  - ship_to_org_id

3.6.7 Arbitrary Parameters

You can use the <param> document element to send arbitrary parameters that are not already provided, or that may be required for particular applications. You would specify the arbitrary parameter as a name-value pair, using the syntax described in Section 3.5.1, "Parameter Syntax" on page 3-4:

```
<param name="parameter_name">parameter_value</param>
```

For example:

```
<param name="org_home_page">http://www.oracle.com</param>
```

Such arbitrary parameters are not processed by the UI Server, but are passed to the Oracle Configuration Interface Object (CIO), thus making them available to Functional Companions. (See the *Oracle Configuration Interface Object (CIO) Developer's Guide* for information about obtaining a list of the initialization parameters passed.)

While the architecture of Oracle Configurator allows for the possibility of validating XML parameters against a DTD, this is not currently enforced.

3.6.8 Parameter Compatibility

The initialization message parameters for this release are backwardly compatible. A host application can continue to use the initialization message parameters provided for the previous release with the same results, unless a parameter has been replaced or withdrawn, thus making it obsolete.

Obsolete parameters are listed in Section 3.6.9, "Obsolete Parameters" on page 3-14.

3.6.9 Obsolete Parameters

Obsolete parameters in the initialization message are ignored by Oracle Configurator. Your host application does not need to remove these parameters from the initialization message, but they will have no effect on the initialization of your application.

- agreement_id
- agreement_type_code
- gsa
- invoice_to_site_use_id
- order_type_id
- po_number
- price_list_id
- responsibility_id

3.7 Initialization Parameter Descriptions

This section lists alphabetically all the parameters of the initialization message. The use of parameters in the initialization message is described in Section 3.5, "Setting Parameters" on page 3-4. The parameters are summarized in Table 3–4.

Table 3–4 Initialization Parameters for Oracle Configurator

Name
alt_database_name
application_id
apps_connection_info
atp_package_name
calling_application_id
config_creation_date
config_effective_date
config_effective_usage
config_header_id
config_model_lookup_date
config_rev_nbr
configurator_session_key
context_org_id

Table 3–4 Initialization Parameters for Oracle Configurator

Name
customer_id
customer_site_id
database_id
fndnam
get_atp_dates_proc
gwyuid
icx_session_ticket
inventory_item_id
model_id
model_quantity
organization_id
price_mult_items_mls_proc
price_mult_items_proc
price_single_item_proc
pricing_package_name
product_id
publication_mode
pwd
read_only
requested_date
return_url
save_config_behavior
ship_to_org_id
template_url
terminate_id
terminate_msg_behavior
two_task

Table 3–4 Initialization Parameters for Oracle Configurator

Name
ui_def_id
user
user_id
warehouse_id

alt_database_name

A fully specified JDBC connect string or URL, specifying the JDBC driver and the database alias of the database to connect to. Recommended for use during development of your application, as an alternative to connecting as an Oracle Applications user. Not recommended for production deployment. Must specify thin drivers, for example: `jdbc:oracle:thin:@server01:1521:vis11`.

application_id

The ID from FND_APPLICATION.APPLICATION_ID that is the id of the host application.

apps_connection_info

If the Oracle Configurator is running in one database (e.g., Release 11*i*), and connecting to another database to perform pricing, this parameter describes how to connect to the other database. The apps_connection_info element can contain one of the following parameters or sets of parameters:

- `database_id`
- `database_id` and `icx_session_ticket`
- `user`, `pwd`, `gwyuid`, `fndnam`, and `two_task`
- `alt_database_name`, `user`, and `pwd`

atp_package_name

The name of the PL/SQL interface package that the OC Servlet will call to get ATP information. This parameter is required, if the ATP callback interface is to be used. The particular procedure in the package to be used for calculating ATP dates is specified by `get_atp_dates_proc`.

calling_application_id

The ID from FND_APPLICATION.APPLICATION_ID that is the ID of the host application

In using Oracle Configurator Developer to publish models, you must select at least one application from the list of all registered applications. Applications that are not part of Oracle Applications should be registered as Oracle Applications before they can be used in Developer. (See the *Oracle Applications Developer's Guide*.)

When the publication is created, a value for FND_APPLICATION.APPLICATION_ID is saved in the database. It is very important to know that if the development and production publications are on separate servers, then the custom application must be registered on both servers; it is your responsibility to verify that the custom application's ID is the same on both servers.

Required.

config_creation_date

The host application's notion of when the configuration is created.

The value for the `config_creation_date` parameter must be determined by your host application. It is the host application's notion of when the configuration was created.

The value must be in the format "MM-DD-YYYY-HH-mm-SS". The values for the elements of this format are shown in the following table:

MM	the number of the month
DD	the number of the day of the month
YYYY	the year
HH	the 24-hour representation of the hour
mm	the number of minutes
SS	the number of seconds

Example `<param name="config_creation_date">03-25-2001-19-30-02</param>`

Defaults For a new configuration: the value of SYSDATE. For a restored configuration: the saved value of `config_creation_date`. If the parameter value does not include the "HH-mm-SS" portion, then the default time is assumed to be midnight ("00-00-00").

config_effective_date

The date used to filter effective nodes and rules.

This parameter has the same structure as `config_creation_date`.

Defaults For a new configuration: the value of `config_creation_date`. For a restored configuration: the saved value of `config_effective_date`.

Not required.

config_effective_usage

The publishing Usage name. The value is not case-sensitive.

Determines the publishing Usage name for the configuration Model. If this parameter is missing, Oracle Configurator will check the Oracle Applications profile option CZ:Usage Name. If this profile option is not defined, the default value of this profile option, which is "Any Usage", is used.

Default The default value is "Any Usage", unless CZ:Usage Name has been set.

Not required.

config_header_id

The identifier for an existing configuration. Only used for retrieving a configuration previously saved by the run-time Oracle Configurator. Not present if the configuration was not saved.

The value for the `config_header_id` parameter is obtained from CZ_CONFIG_HDRS.CONFIG_HDR_ID in the Oracle Configurator schema.

config_model_lookup_date

Date to look up the publication for the configuration Model. This parameter has the same structure as `config_creation_date`.

Defaults For a new configuration: the value of `config_creation_date`. For a restored configuration: the saved value of `config_effective_date`.

Not required.

config_rev_nbr

The configuration revision number. Only used for retrieving a configuration previously saved by the run-time Oracle Configurator. Not present if the configuration was not saved.

The value for the `config_rev_nbr` parameter is obtained from CZ_CONFIG_HDRS.CONFIG_REV_NBR in the Oracle Configurator schema.

configurator_session_key

An application-dependent string that identifies a configuration session, and allows linking a pricing or ATP request from the configuration window to the host application entity that started the configuration session.Examples for creating this key might be: order header ID with order line ID, or quote ID with quote revision number.

context_org_id

The organization identifier for the BOM exploder. The value for the `context_org_id` parameter must be determined by your host application. It is ultimately derived from MTL_SYSTEM_ITEMS.ORGANIZATION_ID.

This parameter should be replaced by its synonym, `organization_id`.

customer_id

When getting ATP dates, the ID of the customer to which the configured product is to be shipped.

customer_site_id

When getting ATP dates, the ID of the customer site to which the configured product is to be shipped.

database_id

The name of a DBC file that contains database connectivity information. This file can be found in a standard Oracle Applications installation by calling the PL/SQL function `fnd_web_config.database_id`. If only the database_id is called, it will use the entries `batch_validate_user` and `batch_validate_password` to find an Oracle Applications username and password to use for the database connection.

fndnam

Used to establish an Oracle Applications context. Its value can be read from the Forms environment variable FNDNAM. Oracle Configurator authenticates standard Oracle Applications users by using the FND login algorithms. Used in conjunction with `gwyuid`.

get_atp_dates_proc

The name of the "Get ATP Dates" procedure to be called from the package specified by `atp_package_name`. This parameter is conditionally required; it must be provided if the ATP callback interface is to be used.

gwyuid

Used to establish an Oracle Applications context. Its value can be read from the Forms environment variable GWYUID. Used in conjunction with `fndnam`.

icx_session_ticket

An icx_session_ticket encodes an Oracle Applications session. You should use the PL/SQL function `cz_cf_api.icx_session_ticket` to obtain a value for this parameter. This is the recommended way for Oracle Applications to call the configuration window. When passing an `icx_session_ticket`, the host application must also pass a `database_id`.

inventory_item_id

This is the synonym parameter for model_id. It is the inventory item identifier for the top Model. It is the key to lookup the configuration model information from CZ_MODELS table.

Conditionally required. No default.

This parameter is a synonym that replaces model_id.

model_id

The inventory item identifier for the top-level Model.

The value for the `model_id` parameter must be determined by your host application. It is ultimately derived from MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID.

Conditionally required. No default.

This parameter should be replaced by its synonym, `inventory_item_id`.

model_quantity

The value of this parameter is a number that indicates how many of the Model are being configured. The model quantity may change during a configuration session, so the final quantity should be read from the associated output item in the termination message.

Default For a new configuration, the default is 1. The host application may set a different number.

organization_id

If you are using Oracle Applications Order Management, this is the organization identifier for the BOM exploder. The value should be the same as OM: Item Validation Organization.If you are using a multiple organization structure, your system administrator must change the *OM: Item Validation Organization* parameter to be visible and updatable at the responsibility level. This change allows Order Management to default code and revenue account information accurately. This is not the warehouse id.

This parameter is a synonym that replaces `context_org_id`.

price_mult_items_mls_proc

This is the name of the "Price Multiple Items" procedure to be called in an MLS environment. This parameter should be used by calling application that supports multiple currencies, not just USD (US dollars).

This parameter is conditionally required; one of this parameter, `price_single_item_proc`, or `price_mult_items_proc` must be provided if pricing callbacks are to be used.

price_mult_items_proc

The name of the "Price Multiple Items" procedure to be called from the package specified by `pricing_package_name`.

This parameter is conditionally required; one of this parameter, `price_single_item_proc`, or `price_mult_items_mls_proc` must be provided if pricing callbacks are to be used.

You should use `price_mult_items_mls_proc` in preference to this parameter, since the procedure called through this parameter displays prices only in USD (US dollars).

This parameter takes precedence over `price_single_item_proc`.

price_single_item_proc

The name of the "Price Single Item" procedure to be called from the package specified by `pricing_package_name`.

This parameter is conditionally required; one of this parameter, price_mult_items_proc, or price_mult_items_mls_proc must be provided if pricing callbacks are to be used.

This procedure will not be called if price_mult_items_proc is provided.

Deprecated This parameter is now deprecated; use price_mult_items_proc if possible.

pricing_package_name

The name of the PL/SQL interface package that the OC Servlet will call to get pricing information. This parameter is required if the pricing callback interface is to be used. The particular procedure in the package to be used for performing pricing is specified by either price_mult_items_proc or price_single_item_proc.

product_id

For imported Models, Inventory_item_id +organization_id. For Models created in Developer, entered by user.

Conditionally required. No default.

publication_mode

Determines the publication mode for the configuration Model. If this parameter is missing, Configurator will check the Oracle Applications profile option CZ:Publication Mode. If this option is not defined, Production mode ("P") will be used to look up the publication.

The values allowed for this parameter are shown in the following table:

P Production

T Test

Default The default value is "P".

Not required.

pwd

The password to use when logging in. Use the Oracle Applications password if you identified the database with the two_task parameter. Use the database password if you identified the database with the alt_database_name parameter. Used in conjunction with user.

read_only

If the value is "true", the UI Server provides a read-only UI for viewing configurations. The end user can examine options, but cannot select any. The "Done" button is disabled. The UI Server displays a message at the beginning of the configuration session, indicating that the session is read-only.

Default "false"

requested_date

When getting ATP dates, the requested date entered on the order line. The format of the date must be "MM-dd-yyyy". The default value of SYSDATE is used if you do not specify a different date.

return_url

The fully qualified URL of a Java servlet installed on your web server that implements the behavior that you want after the user has closed the configuration window. Used only when the run-time Oracle Configurator is Dynamic HTML in a browser. See Section 3.6.4, "Return URL Parameter" on page 3-11 for details.

save_config_behavior

The values allowed for this parameter are shown in the following table:

`never`	A new configuration will not be saved.
`new_config`	A new configuration will be saved.
`new_revision`	A new revision of the configuration will be saved. (If no existing revision is found, a new configuration will be saved.)
`overwrite`	The existing configuration header and revision will be used.

Default "new_revision"

If the value is `overwrite`, an error will be signalled.

ship_to_org_id

When getting ATP dates, the ID of the organization to which the configured product is to be shipped. This value is obtained from SHIP_TO_ORG_ID in the OE_ORDER_LINES_ALL table.

template_url

The URL of the template file that the configuration window will use when displaying its initial state. It is the responsibility of the host application to choose the correct template, if there need to be multiple templates for multiple languages or browsers. The web page pointed to by the template URL must contain the content frame and the proxy frame. You may need to account for language-specific installation directory names, such as `OA_HTML/US`, when specifying this parameter. Used only when the run-time Oracle Configurator is Dynamic HTML in a browser.

Defaults For a Netscape browser: "czFraNS.htm". For an Internet Explorer browser: "czFraIE.htm".

Note: Oracle Configurator provides National Language Support (NLS) (one language at a time), but not Multiple Language Support (MLS) (multiple languages concurrently).

---

terminate_id

Identification number used to support guided selling in Oracle Order Management. An Applet session running in the UI Server generates a termination ID (which is a sequence number) and inserts it into the initialization message for the DHTML session (also running in the UI Server), as the value of this initialization parameter. When the DHTML session terminates, it stores its XML termination message in the database, identified by this termination ID. The Applet session then uses the termination ID to fetch the XML termination message from the database and return it to the host application (Order Management). For a related subject, see the discussion of the "heartbeat" mechanism and guided selling in the *Oracle Configurator Installation Guide*.

terminate_msg_behavior

The values allowed for this parameter are shown in the following table:

`full`	The entire termination message is passed back to the host application. This includes prices, if you have used a pricing interface package (see Chapter 7).
`brief`	No output or messages are passed to the caller.

It is recommended that host applications using the CZ_CONFIG_DETAILS_V view to read configuration outputs use `brief` when the configuration is saved. If the

configuration is not saved, then the outputs and messages will not be readable from the database. If the configurator gets a `connection_error` or other error, the error messages that it receives will be passed back as messages even if the `terminate_msg_behavior` is brief.

two_task

The name of the database instance to connect to. This value can be read from the environment variable TWO_TASK.

ui_def_id

The identifier for the User Interface created in Configurator Developer. The value for the `ui_def_id` parameter is obtained from CZ_UI_DEFS.UI_DEF_ID in the Oracle Configurator schema. The value can also be obtained by calling the PL/SQL function `cz_cf_api.ui_for_item`.

ui_type

Type of client. The only supported values are "Applet" and "DHTML". Must be set to "DHTML" when using the Oracle Configurator configuration window in a custom web deployment.

user

The username to use when logging in. Use the Oracle Applications username if you identified the database with the `two_task` parameter. Use the database username if you identified the database with the `alt_database_name` parameter. Used in conjunction with `pwd`.

user_id

The ID from FND_USER.USER_ID.

warehouse_id

When getting ATP dates, the ID of the organization that is going to ship the configured product to the customer. This value is obtained from SHIP_FROM_ORG_ID in the OE_ORDER_LINES_ALL table.

4

Session Termination

4.1 How it Works

See Section 3.2, "How it Works" on page 3-1, in Chapter 3, "Session Initialization".

4.2 What You Do

See Section 3.3, "What You Do" on page 3-3, in Chapter 3.

4.3 Definition of Session Termination

Session termination takes place when the configuration window is closed by one of the conditions listed in Table 4–1:

Table 4–1 Termination conditions

Condition	Example	Explanation
Submission	Your user clicks the Done button.	See Section 4.5, "Submission" on page 4-3
Cancellation	Your user clicks the Cancel button.	See Section 4.6, "Cancellation" on page 4-8
Error	A connection cannot be made to the database.	See Section 4.7, "Error" on page 4-9

When the configuration window is closed, terminating your user's configuration session, the OC Servlet returns the results to your host application in the form of a termination message, written in XML. You need to understand the structure of the termination message in order to be able to extract the desired data from it in your

return URL servlet. The structure of this message is described in Section 4.4, "XML Message Structure" on page 4-2.

4.4 XML Message Structure

All outputs in the XML termination message are written as XML elements and subelements of the `<terminate>` document element, in the general form:

```
<terminate>

  <element_name>element_value</element_name>

  <element_name>
    <subelement_name>subelement_value</subelement_name>
  </element_name>

</terminate>
```

The top-level structure of the `<terminate>` element is illustrated by these excerpts from its DTD:

```
...
<!ELEMENT terminate (config_header_id?, config_rev_nbr?, valid_configuration?,
complete_configuration?, exit, config_outputs?, config_messages?)>
...
<!ELEMENT config_outputs (output_option*)>
...
<!ELEMENT config_messages (message*)>
...
```

Example 4–1 shows the basic structure of a sample XML termination message. Typographical emphasis and comments have been added to point out the structure; such comments do not appear in actual termination messages.

Example 4–1 Example of structure of termination message

```
<terminate>
  <!-- configuration status elements -->
  <config_header_id>1780</config_header_id>
  <config_rev_nbr>2</config_rev_nbr>
  <valid_configuration>true</valid_configuration>
  <complete_configuration>true</complete_configuration>
  <exit>save</exit>
  <config_outputs>
    <option>
```

```
      <component_code>143-1490</component_code>
      <quantity>1</quantity>
      <list_price>0.00</list_price>
      <!-- more elements go here -->
    </option>
    <!-- more options go here -->
  </config_outputs>
  <config_messages>
    <message>
      <message_type>error</message_type>
      <message_text>Config header does not exist in database.</message_text>
    </message>
    <!-- more messages go here -->
  </config_messages>
</terminate>
```

4.5 Submission

Submission occurs after your user closes the configuration window by clicking the “Done” button.

The meaning of the Done button is defined by the context of your host application. For instance, in a web store, it might mean adding the configured product to your user’s “shopping cart”, or submitting the configured order to your order entry system.

The Done button can be customized, as described in Section 6.1.2, "The Header Frame" on page 6-4, in Chapter 6, "User Interface".

When the Done button is clicked, the OC Servlet determines whether a return URL has been specified. If so, the servlet it identifies is executed, and the results it generates are passed to your host application for further processing. This is the most important job of the return URL servlet; it captures the configuration selections of your user so that your host application can make use of them. For more details, see Section 4.8, "The Return URL" on page 4-9

After the configuration window is closed, your host application must repaint the frame used by the configuration window.

After submission, the termination message provides the host application with data describing:

- Section 4.5.1, "Configuration Status"
- Section 4.5.2, "Configuration Outputs"

- Section 4.5.3, "Configuration Messages"

Note: If you are providing guided selling in Oracle Applications Order Management, then your host application should obtain the termination message by using the initialization parameter terminate_id. See the description of that parameter for details.

4.5.1 Configuration Status

The current configuration status is described by the subelements of <terminate> listed in this section.

Subelements

config_header_id

The main identifier of an existing configuration. See the description for config_header_id on page 3-19. This value is displayed in the configuration window with the default label “Configuration Header ID”.

config_rev_nbr

The revision number of an existing configuration. See the description for config_rev_nbr on page 3-19. This value is displayed in the configuration window with the default label “Configuration Revision”.

valid_configuration

The value will be “true” if no error messages are reported for the configuration. This value is displayed in the configuration window with the default label “Configuration Valid”.

complete_configuration

The value will be “true” if all mandatory option classes (required features) are satisfied. This value is displayed in the configuration window with the default label “Configuration Complete”.

exit

The possible values written for this element are shown in the following table:

save	If the configuration was saved.

`cancel`	If the configuration was cancelled.
`error`	If an error was detected while executing in the UI Server.
`processed`	If a batch validation message was processed but not saved.

This value is displayed in the configuration window with the default label “Exit Status”.

prices_calculated_flag

Prices are calculated when the user clicks the “Summary” button. This element tells the host application whether this calculation has happened in synchronization with the configuration. The possible values written for this element and their meanings are shown in the following table:

`true`	The configuration has not been changed since the end user clicked the “Summary” button. That is, the calculated prices are still in synchronization with the configuration.
`false`	Prices were not calculated after the configuration had been changed. This could happen if the end user had never clicked the “Summary” button before clicking “Done”, or if the user changed the configuration and did not click the “Summary” button before clicking “Done”. In this case, the host application should reprice each configuration line, to ensure that the proper prices are applied to the configuration.

4.5.2 Configuration Outputs

The list of options selected by your user during the configuration session is contained in the `<config_outputs>` subelement of `<terminate>`. Each option is enclosed in `<option>` tags and contains the elements described in this section.

Example 4–2 shows an example of configuration outputs in the termination message, with typographical emphasis and comments added.

Example 4–2 Configuration outputs in the termination message

```
<terminate>
  <!-- configuration status goes here -->
  <config_outputs>
    <option>
      <selection_line_id>1846</selection_line_id>
```

```
      <parent_line_id>1847</parent_line_id>
      <component_code>143-1490</component_code>
      <quantity>1</quantity>
      <list_price>0.00</list_price>
      <inventory_item_id>1490</inventory_item_id>
      <organization_id>204</organization_id>
      <uom>Ea</uom>
      <discounted_price>0.00</discounted_price>
      <atp_date></atp_date>
    </option>
    <!-- more options go here -->
  </config_outputs>
  <!-- configuration messages go here -->
</terminate>
```

Subelements

selection_line_id

Contains the ID of the configuration line. It is the same as CZ_CONFIG_ITEMS.CONFIG_ITEM_ID in the Oracle Configurator schema.

component_code

Contains a value extracted from BOM_EXPLOSIONS.COMPONENT_CODE.

parent_line_id

Contains the value from CZ_CONFIG_ITEMS.CONFIG_ITEM_ID for the parent node. It will be “0” for the root.

quantity

Contains the selected quantity for the option.

inventory_item_id

Contains the ID for the item, extracted from MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID.

organization_id

Contains the organization ID for the item, extracted from MTL_SYSTEM_ITEMS.ORGANIZATION_ID

uom

Contains the unit of measure.

atp_date

Contains the ATP date. This is calculated by using the ATP procedure specified in the initialization message. See Section 3.6.6, "ATP Parameters" on page 3-13, and Chapter 7, "Pricing and ATP" on page 7-1.

list_price

Contains the list price for the selected option. This is calculated by using the pricing procedure specified in the initialization message. See Section 3.6.5, "Pricing Parameters" on page 3-12, and Chapter 7, "Pricing and ATP" on page 7-1.

discounted_price

Contains the discounted price for the selected option. This is calculated by using the pricing procedure specified in the initialization message. See Section 3.6.5, "Pricing Parameters" on page 3-12, and Chapter 7, "Pricing and ATP" on page 7-1.

4.5.3 Configuration Messages

The messages generated by the OC Servlet in response to selections made by your user during the configuration session are contained in the `<config_messages>` subelement of <terminate>. Each message is enclosed in `<message>` tags and contains the elements described in this section.

If there were validation failures during your user's configuration session, each failure on the list of the validation failure objects is returned as a `<message>` element describing the failure. Information about the failure is returned to the OC Servlet as an object of type `oracle.apps.cz.cio.ValidationFailure`, which you can access through the Oracle Configuration Interface Object (CIO); see the Oracle CIO Help for details.

Example 4–3 shows an example of a configuration message in the termination message, with typographical emphasis and comments added.

Example 4–3 Configuration messages in the termination message

```
<terminate>
  <!-- configuration status goes here -->
  <!-- configuration outputs go here -->

  <config_messages>
```

```
    <message>
      <message_type>error</message_type>
      <message_text>Config header does not exist in database.</message_text>
    </message>
    <!-- more messages go here -->
  </config_messages>

</terminate>
```

Subelements

component_code

ps_node_id
If present, one of these elements contains the identifier of the option to which this message is related. May be absent, if the message was not generated by a node.

item_name

Contains the name of the option to which this message is related;

message_type

Contains the severity level of the message; possible values are “suggestion”, “warning”, “overridable error”, “error”, “autoselection”, “autoexclusion”, and “not satisfied”.

message_text

Contains the text of the message.

4.6 Cancellation

Cancellation occurs after your user closes the configuration window by clicking the Cancel button. Control is returned to the host application, and no configuration information is returned. The termination message contains only the `<exit>` subelement, with a value of "cancel":

```
<terminate>
  <exit>cancel</exit>
</terminate>
```

4.7 Error

Error occurs after some condition prevents initialization of the configuration window, or submission of the user's selections. Such conditions might include:

- Incorrect database connection or user login parameters (see Section 3.6.1, "Connection Parameters" on page 3-9)
- Lack of any configuration parameters (see Section 3.6.3, "Model Identification Parameters" on page 3-9)
- Incorrect type for a parameter

If there were validation failures during your user's configuration session, each failure on the list of the validation failure objects is returned as a `<message>` element describing the failure. Information about the failure is returned to the OC Servlet as an object of type `oracle.apps.cz.cio.ValidationFailure`, which you can access through the Oracle Configuration Interface Object (CIO); see the Oracle CIO Help for details.

Control is returned to the host application, and no configuration information is returned. Any validation failures are returned as messages in the `<config_messages>` element. The termination message contains the `<exit>` subelement, with a value of "error":

```
<terminate>
  <exit>error</exit>
</terminate>
```

4.8 The Return URL

The Java servlet specified by the return URL initialization parameter (`return_url`) determines how your host application will use the configuration information produced by your user's selections during a session in the configuration window. The return URL servlet (or just "return URL") is executed upon termination of a configuration session, if you have specified the return URL in your initialization message for the configuration window.

The termination message is passed to the return URL as the value of the XMLmsg argument. The initialization message that was passed to the configurator is also passed to the return URL, as the value of the INITmsg parameter.

The return URL must perform all middle-tier and database processing of the configuration and then return HTML that will close the configuration window and continue with the program flow for the host application.

4.8.1 Specifying the Return URL

You specify the identity of your return URL in the XML initialization message, as the value of the parameter `return_url`:

```
<param name="return_
url">http://www.mysite.com:10130/configurator/Checkout</param>
```

The example parameter above comes from Example 3–3, "Minimal HTML for invoking the configuration window" on page 3-7.

See also:

- Section 3.6.4, "Return URL Parameter" on page 3-11
- return_url on page 3-24
- Section 3.5.1, "Parameter Syntax" on page 3-4

4.8.2 Implementing the Return URL

See Example A–1 in Appendix A for an example of a return URL servlet. You can modify this servlet code for the needs of your host application.

In order to use some of the configuration information returned in the termination message (for instance, the outputs described in Section 4.5.2, "Configuration Outputs" on page 4-5), you can write a method that obtains the value of an element in the termination message by using the `getTagValue()` method of the Checkout servlet.

5

Batch Validation

5.1 How it Works

Batch validation allows a host application to perform tasks such as:

- Validating a configuration in the background
- Determining a configuration quantity
- Deleting lines from a configured order while keeping the configuration valid
- Re-validating a previously booked order, if the configuration rules have changed in the meantime
- Using a user interface other than those provided with Oracle Configurator (namely the Java applet and DHTML interfaces).

Batch validation can be called either by:

- passing the batch validation message to the URL of the OC Servlet
- calling the CZ_CF_API.VALIDATE PL/SQL procedure

If the host application is written in PL/SQL, it should call the VALIDATE procedure. There are restrictions in the way that PL/SQL can request data from a URL that require PL/SQL programs to use the cz_cfg_api.validate procedure.

5.2 Passing the Batch Validation Message

A batch validation message consists of information defining the configuration context (such as an identifier for the configured model) and a list of configured options. The message can be used to revalidate a previously saved configuration.

A `<batch_validate>` element is composed of an `<initialize>` subelement and a `<config_inputs>` subelement.

The <initialize> element is described in Chapter 3, "Session Initialization". The parameters of the initialization message are described in Section 3.7, "Initialization Parameter Descriptions" on page 3-15. See the description of the database_id parameter on page 3-20 for connectivity information.

The <config_inputs> element consists of a list of <option> elements, where the <option> element is described in Chapter 4, "Session Termination".

When an <option> element is used in a <config_inputs> element, only the <component_code> and <quantity> elements of the <option> will be used.

Example 5–1 Example of Batch Validation Message

```
<batch_validate>
  <initialize>
    <param name="context_org_id">204</param>
    <param name="config_creation_date">1999/11/23</param>
    <param name="calling_application_id">300</param>
    <param name="responsibility_id">20559</param>
    <param name="config_header_id">21361</param>
    <param name="config_rev_nbr">1</param>
    <param name="read_only">FALSE</param>
    <param name="save_config_behavior">new_revision</param>
    <param name="database_id">ap115sun_dev115</param>
  </initialize>
  <config_inputs>
    <option>
      <component_code>143-1490-1494</component_code>
      <quantity>1</quantity>
    </option>
    <option>
      <component_code>143-297</component_code>
      <quantity>1</quantity>
    </option>
  </config_inputs>
</batch_validate>
```

5.3 Calling the CZ_CF_API.VALIDATE Procedure

Table 5–1 describes the parameters for the CZ_CF_API.VALIDATE procedure.

Table 5–1 Parameters for the CZ_CF_API.VALIDATE procedure

Parameter	Data Type	In/Out	Note
config_input_list	CFG_INPUT_LIST	IN	A table of selections to change. the table contains a series of INPUT_ SELECTION records, where each INPUT_SELECTION contains the fields: COMPONENT_CODE of type Varchar2(1200) QUANTITY of type Number
init_message	VARCHAR2	IN	The initialization message.
config_messages	CFG_OUTPUT_PIECES	OUT	The result of the batch validation message. This data type is actually just UTL_HTTP.HTML_PIECES, a table of type Varchar2(2000).
validation_status	Number	OUT	0 if the call succeeds and config_ messages contains a terminate message. Possible values that can be returned are: 0 CONFIG_PROCESSED 1 CONFIG_PROCESSED_NO_ TERMINATE 2 INIT_TOO_LONG 3 INVALID_OPTION_REQUEST 4 CONFIG_EXCEPTION 5 DATABASE_ERROR 6 UTL_HTTP_INIT_FAILED 7 UTL_HTTP_REQUEST_FAILED
URL	VARCHAR2	IN	Defaulted to be FND_ PROFILE.Value('CZ_UIMGR_URL')

5.4 Examples

Example 5–2 and Example 5–3 show fragments from a PL/SQL program that calls the CZ_CF_API.VALIDATE procedure

Example 5–2 Example 1 of Calling the CZ_CF _API.VALIDATE Procedure

```
...
/*---------------------------------------------------------------------------
Procedure Name : Send_input_XML
Description    : sends the xml batch validation message to hostapp that has
                 options that are newly inserted/updated/deleted
                 from the model.
---------------------------------------------------------------------------*/

PROCEDURE Send_input_XML
            ( p_model_line_id       IN NUMBER ,
              p_org_id              IN NUMBER ,
              p_model_id            IN NUMBER ,
              p_config_header_id    IN NUMBER ,
              p_config_rev_nbr      IN NUMBER ,
              p_model_qty           IN NUMBER ,
              p_creation_date       IN DATE ,
              p_deleted_options_tbl IN   App_Order.request_tbl_type
                                    := App_Order.G_MISS_REQUEST_TBL,
              p_updated_options_tbl IN   App_Order.request_tbl_type
                                    := App_Order.G_MISS_REQUEST_TBL,
              x_out_XML_msg         OUT LONG ,
              x_return_status       OUT VARCHAR2 )
...
      l_html_pieces                  CZ_CF_API.CFG_OUTPUT_PIECES;
      l_option                       CZ_CF_API.INPUT_SELECTION;
      l_batch_val_tbl                CZ_CF_API.CFG_INPUT_LIST;
...
       Create_hdr_XML
            ( p_model_line_id        => p_model_line_id ,
              p_org_id               => p_org_id ,
              p_model_id             => p_model_id ,
              p_config_header_id     => p_config_header_id ,
              p_config_rev_nbr       => p_config_rev_nbr ,
              p_model_qty            => p_model_qty ,
              p_creation_date        => p_creation_date ,
              x_XML_hdr              => l_XML_hdr);


...
CZ_CF_API.Validate( config_input_list => l_batch_val_tbl ,
                          init_message      => l_XML_hdr ,
                          config_messages   => l_html_pieces ,
                          validation_status => l_validation_status ,
                          URL               => l_url );
```

Example 5–3 Example 2 of Calling the CZ_CF _API.VALIDATE Procedure

```
set serveroutput on
set verify off

-- Run this query in SQL*Plus, providing input of model id
-- This query is like what the calling application might send.
-- The output might go back to some other servlet.



BEGIN
declare
  config_input_list  CZ_CF_API.CFG_INPUT_LIST;
  ---- OC Servlet URL needs to be entered here....
  l_url varchar2(100):=
'http://www.mysite.com:10130/configurator/oracle.apps.cz.servlet.UiServlet' ;
  init_message varchar2(4000):='<initialize>';
  config_messages  CZ_CF_API.CFG_OUTPUT_PIECES;
  validation_status  NUMBER;
  list_indx number := 1 ;

  begtime varchar2(30) := null ;
  endtime varchar2(30) := null ;


--- Build the initialization message.
        TYPE param_name_type IS TABLE OF VARCHAR2(25)
                          INDEX BY BINARY_INTEGER;
        TYPE param_value_type IS TABLE OF VARCHAR2(40) -- 40, to fit alt_
database_name
                          INDEX BY BINARY_INTEGER;



        param_name      param_name_type;
        param_value     param_value_type;
        l_rec_index         BINARY_INTEGER;

        l_context_org_id               VARCHAR2(30);
        l_config_creation_date         VARCHAR2(30);
        l_two_task                     VARCHAR2(30);
        l_user                         VARCHAR2(30);
        l_pwd                          VARCHAR2(30);
        l_fndnam                       VARCHAR2(30);
```

```
        l_calling_application_id        VARCHAR2(30);
        l_responsibility_id             VARCHAR2(30);
        l_model_id                      VARCHAR2(30);
        l_config_header_id              VARCHAR2(30);
        l_config_rev_nbr                VARCHAR2(30);
        l_gwyuid                        VARCHAR2(30);
        l_read_only                     VARCHAR2(30);
        l_save_config_behavior          VARCHAR2(30);
        l_save_usage_behavior           VARCHAR2(30);
        l_ui_type                       VARCHAR2(30);
        l_so_line_id                    VARCHAR2(30);
        l_validation_org_id             VARCHAR2(30);
        l_dbc                           VARCHAR2(30);
        l_model_quantity                VARCHAR2(30);
        l_termination                   VARCHAR2(30);
    l_alt_database_name                 VARCHAR2(40);  -- note extra length

--Options

        l_component_code                VARCHAR2(2000);
        l_option_quantity               VARCHAR2(30);
        l_test_param                    VARCHAR2(20);

BEGIN

        param_name(1) := 'context_org_id';
        param_name(2) := 'config_creation_date';
        param_name(3) := 'two_task';
        param_name(4) := 'user';
        param_name(5) := 'pwd';
        param_name(6) := 'fndnam';
        param_name(7) := 'calling_application_id';
        param_name(8) := 'responsibility_id';
        param_name(9) := 'model_id';
        param_name(10) := 'config_header_id';
        param_name(11) := 'config_rev_nbr';
        param_name(12) := 'gwyuid';
        param_name(13) := 'read_only';
        param_name(14) := 'save_config_behavior';
        param_name(15) := 'save_usage_behavior';
        param_name(16) := 'model_quantity';
        param_name(17) := 'database_id';
        param_name(18) := 'terminate_msg_behavior';
        param_name(19) := 'alt_database_name';
```

```
SELECT
        '204',   -- corrected value
        '10-16-2000-09-41-12',
        null,
        null,
        null,
        null,
        '660',
        '50171',
        '&modelId',  -- 143 is the usual value
        null,
        null,
        null,
        null,
        'new_revision',
        null,
        '45',
        'ap505dbs_dom1151',
        'brief',
        'jdbc:oracle:thin:@serv01:1521:sid02'
INTO
        l_context_org_id,
        l_config_creation_date,
        l_two_task,
        l_user,
        l_pwd,
        l_fndnam,
        l_calling_application_id,
        l_responsibility_id,
        l_model_id,
        l_config_header_id,
        l_config_rev_nbr,
        l_gwyuid,
        l_read_only,
        l_save_config_behavior,
        l_save_usage_behavior,
        l_model_quantity,
        l_dbc,
        l_termination,
        l_alt_database_name
FROM    dual ;


param_value(1) := l_context_org_id;
```

```
        param_value(2) := l_config_creation_date;
        param_value(3) := l_two_task;
        param_value(4) := l_user;
        param_value(5) := l_pwd;
        param_value(6) := l_fndnam;
        param_value(7) := l_calling_application_id;
        param_value(8) := l_responsibility_id;
        param_value(9) := l_model_id;
        param_value(10) := l_config_header_id;
        param_value(11) := l_config_rev_nbr;
        param_value(12) := l_gwyuid;
        param_value(13) := l_read_only;
        param_value(14) := l_save_config_behavior;
        param_value(15) := l_save_usage_behavior;
        param_value(16) := l_model_quantity;
        param_value(17) := l_dbc;
        param_value(18) := l_termination;
        param_value(19) := l_alt_database_name;

        l_rec_index := 1;


        LOOP

                IF (param_value(l_rec_index) IS NOT NULL) THEN


                init_message  := init_message || '<param name=' ||    '"'    ||
param_name(l_rec_index) ||   '"'   ||'>'||
                                   param_value(l_rec_index)||'</param>';


                END IF;


                EXIT WHEN l_rec_index > 18;  -- adjust for number of parameters
                l_rec_index := l_rec_index + 1;

        END LOOP;


        init_message := init_message || '</initialize>';
        init_message := REPLACE(init_message,' ','+');
```

```
        dbms_output.enable(buffer_size => 200000);
                dbms_output.put_line(substr(init_message,1,255));
                dbms_output.put_line(substr(init_message,256,255));
                dbms_output.put_line(substr(init_message,512,255));
                dbms_output.put_line(substr(init_message,768,255));
                dbms_output.put_line(substr(init_message,1024,255));
                dbms_output.put_line(substr(init_message,1280,255));


  CZ_CF_API.VALIDATE(config_input_list,init_message,config_messages,validation_
status,l_url);

  IF(validation_status=CZ_CF_API.CONFIG_PROCESSED)THEN
    dbms_output.put_line('Config processed successfully');
  ELSIF(validation_status=CZ_CF_API.CONFIG_PROCESSED_NO_TERMINATE)THEN
    dbms_output.put_line('Config processed successfully, no termination
message');
  ELSIF(validation_status=CZ_CF_API.INIT_TOO_LONG)THEN
    dbms_output.put_line('Init message too long');
  ELSIF(validation_status=CZ_CF_API.INVALID_OPTION_REQUEST)THEN
    dbms_output.put_line('Invalid option request');
  ELSIF(validation_status=CZ_CF_API.CONFIG_EXCEPTION)THEN
    dbms_output.put_line('General config exception');
  ELSIF(validation_status=CZ_CF_API.DATABASE_ERROR)THEN
    dbms_output.put_line('Database error');
  ELSIF(validation_status=CZ_CF_API.UTL_HTTP_INIT_FAILED)THEN
    dbms_output.put_line('UTL_HTTP: initialization failed');
  ELSIF(validation_status=CZ_CF_API.UTL_HTTP_REQUEST_FAILED)THEN
    dbms_output.put_line('UTL_HTTP: request failed');
  ELSE
    dbms_output.put_line('Unknown error');
  END IF;
        l_rec_index := config_messages.FIRST;
        dbms_output.put_line ( 'Recieved Response from the server follows ....'
);

        LOOP
                dbms_output.put_line(  ltrim(rtrim(substr(config_messages(l_rec_
index),1,255))));
                dbms_output.put_line(  ltrim(rtrim(substr(config_messages(l_rec_
index),256,255))));
                dbms_output.put_line(  ltrim(rtrim(substr(config_messages(l_rec_
index),512,255))));
                dbms_output.put_line(  ltrim(rtrim(substr(config_messages(l_rec_
index),768,255))));
```

```
                dbms_output.put_line( ltrim(rtrim(substr(config_messages(l_rec_
index),1024,255))));
                dbms_output.put_line( ltrim(rtrim(substr(config_messages(l_rec_
index),1280,255))));
                dbms_output.put_line( ltrim(rtrim(substr(config_messages(l_rec_
index),1536,255))));
                dbms_output.put_line( ltrim(rtrim(substr(config_messages(l_rec_
index),1792))));

        EXIT WHEN l_rec_index = config_messages.LAST;
                l_rec_index := config_messages.NEXT(l_rec_index);

        END LOOP;

        dbms_output.put_line ('Servlet URL used follows ....');
    dbms_output.put_line(ltrim(rtrim(l_url)));


END;
END;
/
```

6

User Interface

Oracle Configurator includes a UI Server that renders a DHTML representation of an Oracle Configurator Model and user interface, as defined in Configurator Developer and published in the Active Model and Active UI. This document does not explain how to customize a custom user interface not generated through Configurator Developer, or using a non-Oracle host application.

Oracle Configurator provides a set of HTML files that make up the parts of a single HTML Template. This Template can be used as a model for producing alternate Template designs. This enables you to incorporate the configuration window into other host applications, retaining the functionality built into the configuration window while adapting its look and feel to match the surrounding frames of the host application.

Custom Template versions can be created in any HTML editor or text editor.

6.1 How it Works

The HTML Template has a structure that supports the operation of the configuration window.

6.1.1 Structure of the HTML Template

It is essential to understand the basic structure of the HTML Template, in order to know which parts you can customize, and how.

The Template is divided into two areas:

- required template elements, which are essential for supporting the operation of the DHTML components
- optional template elements, which you can customize and augment

The required structure must be incorporated into every custom Template.

Figure 6–1 on page 6-2 shows the structure of the template, which consists of:

- the optional Header Frame
- the required Content Frame, which is split into a Tree view and a Display view
- the required Source, Proxy, and Heartbeat Frames

Figure 6–1 The Structure of the HTML Template

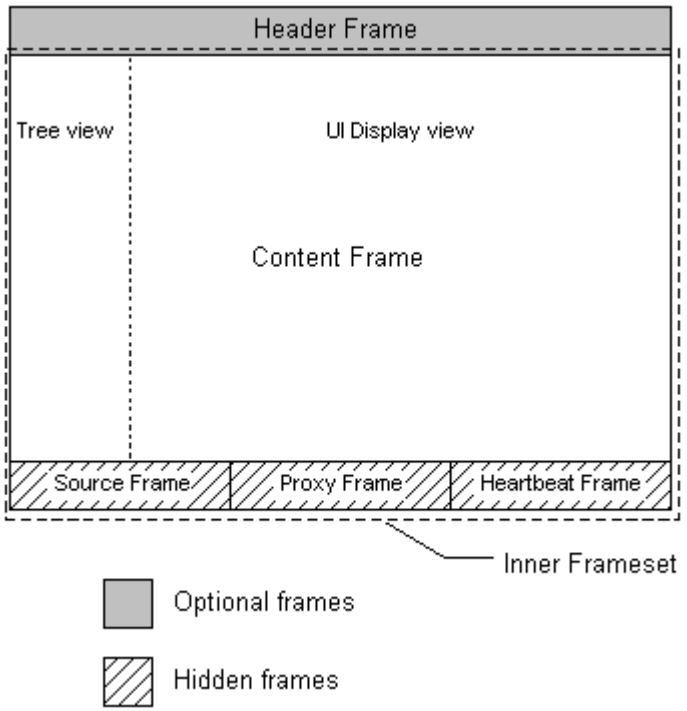


Table 6–1 lists the constraints on customizing the Template, and the files in which the elements of the Template are implemented.

Table 6–1 Constraints on the HTML Template

Required?	Customize?	Frame/Frameset				File
No	Yes	Outer Frameset				czFraIE.htm czFraNS.htm
No	Yes			Header Frame		czHeader.htm
Yes	No		Inner Frameset	Hidden Frameset		czFraTemplate.htm
Yes	No				Source Frame	czSource.htm
Yes	No				Proxy Frame	Proxy.class (servlet)
Yes	No				Heartbeat Frame	czHeartbeat.htm
Yes	No			Content Frame		czCntnt.htm
Yes	No			Tree view		cztree.htm
Yes	No			Display view		czdisp.htm

The framesets are defined in the file czFraTemplate.htm.

The Content Frame is the location of the configuration window, including the summary. The Source Frame contains the JavaScript controls that define the behavior of the DHTML user interface. The Proxy Frame is the communication center between the configuration window and the UI Server. The Heartbeat Frame is used to communicate with the UI Server when the DHTML interface is used for guided selling in an Oracle Applications context (see the *Oracle Configurator Installation Guide* for more information).

There are no controls for submitting and canceling configurations, in this structure. It is up to the implementer to put the controls for triggering functions in the Template or in the host application. Triggering functions are:

- submit
- cancel
- show configuration or summary information
- display ATP information

The Source, Proxy, and Heartbeat Frames are hidden from the end user, by setting their height allocations to 0 (zero) in the Inner Frameset, which contains them. The Inner Frameset is defined in the browser-specific frameset files listed in Table 6–2.

Table 6–2 Browser-Specific Frameset Files

Browser	Frameset File	Java System Property
Netscape	czFraNS.htm	cz.uiservlet.templateurl
Internet Explorer	czFraIE.htm	cz.uiservlet.templateurl.ie

If you substitute different frameset files, you need to set the corresponding Java System Properties accordingly. For information on how to set these parameters, see the reference section on Java System Property Parameters for the UI Servlet in the *Oracle Configurator Installation Guide*. You can also specify the frameset file in the initialization message for the OC Servlet, using the parameter template_url, described on page 3-25.

6.1.2 The Header Frame

This frame loads the HTML page **czHeader.htm** (Configurator Header). It consists of your choice of GIFs to display the desired header graphics, and a set of image buttons, listed in Table 6–3 on page 6-5.

Figure 6–2 on page 6-4 shows the default Header Frame. It is populated with GIFs and image buttons, which are described in Table 6–3 on page 6-5.

Figure 6–2 The czHeader frame

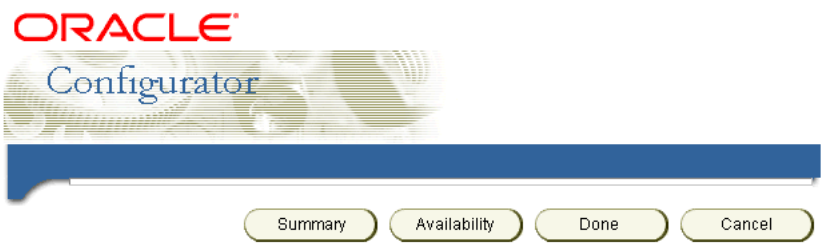


Table 6–3 Buttons in the Header Frame

Button Name	Description
Summary / Configuration	This button is scripted to toggle the Contents Frame between the Oracle Configurator UI and the Summary screen. The script also changes the caption on the button to read "Summary" when the Oracle Configurator UI is displayed and "Configuration" when the Summary screen is displayed.
Availability	This button is scripted to populate the current display with ATP dates. When triggered for the Summary screen, it refreshes the entire Summary page with data that includes the ATP information.
Done	This button is scripted to save the configuration, terminate the session, and return control to the host application.
Cancel	This button is scripted to terminate the session and return control to the host application without saving the configuration.

If you need to add your own buttons, you can add functions to those defined in czHeader.htm. Do not modify the existing functions, or the configuration window may not operate correctly.

6.2 What You Do

- You can customize the decorations in the Header frame, and perhaps in the Left and Right frames.
- You can customize the buttons in the Header frame.
- You can add new buttons in the Header frame.
- You do *not* modify the Content Frame, since its contents are automatically generated by the OC Servlet.

6.2.1 The Default Configuration Window

You normally do not have to define a user interface for the configuration window; a default one may be already available.

The layout, navigational model, controls, and other objects displayed in the configuration window are derived from a User Interface definition in the Oracle Configurator schema. The User Interface definition is created from a Model

structure in the Oracle Configurator Developer. The run-time Oracle Configurator can be further customized in Configurator Developer.

The run-time Oracle Configurator displayed in the your web browser presents a configuration Model. Once selections have been made by your user, the configuration window presents the resulting configuration for that Model. Unsatisfied selections are indicated by a customizable icon. By default, the icon is a red asterisk.

6.2.2 Customizing the User Interface in Oracle Configurator Developer

You generate and modify the default UI in Oracle Configurator Developer. Your design is displayed in the UI display view of the Content Frame.

If you want to call Functional Companions from your user interface, define controls to call them, in Configurator Developer.

See the *Oracle Configurator Developer Tutorial* and Oracle Configurator Developer User's Guide.

6.2.3 Restrictions on the Configuration Window

- You cannot use BMP (Windows bitmap) files in your user interface for the configuration window, since this file format is not compatible with rendering in a DHTML context. This will affect you if the User Interface that you defined in Oracle Configurator Developer included any BMP files.

  The configuration window can use GIF, JPG, and other formats compatible with web browsers.

6.3 Customizing the HTML Templates

The template that you are most likely to modify is the Header Frame.

You may also want to modify the frameset files for the Netscape and Internet Explorer browsers (czFraNS.htm and czFraIE.htm, respectively). See Table 6–2 on page 6-4.

Warning: If you modify any of the template files, be sure to make backups. The template files are overwritten whenever you reinstall Oracle Configurator.

The modifiable template files are commented for your guidance.

6.3.1 Specifying the Location of Media Files

When you design a User Interface in Configurator Developer, you can specify the location of custom image media files for your host application: icons, backgrounds, button shapes, and the like. When your host application uses the DHTML configuration window, the UI Server strips the path information (which is stored in the database) from the name of these media files, and uses only the filename itself.

The OC HTML template files set a global variable IMAGESPATH, pointing to the location of the media files, that is read by the JavaScript controls that populate the configuration window. By default, IMAGESPATH is set to /OA_MEDIA/, which is located under the document root directory for your servlet. The web server serves HTML documents from the document root directory, unless it is configured to search elsewhere.

Oracle Configurator stores its own standard media files in OA_MEDIA. Here is a fragment from czHeader.htm , showing the location of an image file for the Done button (emphasized):

```
<!-- done button -->
    <a href="javascript:doneClick();">
      <img name="IMG-done" src="/OA_MEDIA/czDone0.gif" border="0" WIDTH="37"
HEIGHT="15">
    </a>
```

If you have custom media files to use, you can store them in OA_MEDIA. If you wish to use a different location, it is recommended that you add a directory under OA_MEDIA, then modify certain HTML template files to reflect that location.

1. Change this code in czSource.htm:

```
//Modify it depending on your server installation directory.
//** Do not forget to put the trailing SLASH at the end
var IMAGESPATH = '/OA_MEDIA/alt/';
var SOURCEPATH = '/OA_HTML/';
```

2. Change this code in czHeader.htm:

```
/* Global string storing physical or logical path to server
     graphics. */
  var IMAGESPATH = '/OA_MEDIA/alt/';
```

3. Copy your custom media files, and the OC media files, to the new directory.

6.3.2 Modifying the Header Bar

As a simple example of the modifications you can make directly to the HTML template files, try changing the text of the header bar of the configuration window.

Locate the following lines in /OA_HTML/US/czHeader.htm:

```
<div id=prdName class=prd>
<!-- Product name should be given here -->
Configurator
</div>
```

The typographical emphasis indicates what you will change.

Now change the label text as shown:

```
<!-- Product name should be given here -->
My Own Configurator
</div>
```

The changes will take effect when you refresh the web browser displaying the OC Servlet.

6.3.3 Hiding the Model Tree

By default, the Content Frame displays a tree view of the configuration Model next to a view of the User Interface that was defined in Configurator Developer. If you want to hide the tree view, you can also do so by setting a Java System Property Parameter to be passed to the OC Servlet. The parameter is `cz.frameset.allocations.top`. Its default setting is:

```
cz.frameset.allocations.top=30%,*
```

To hide the tree, set the width of the tree view to 0 (zero):

```
cz.frameset.allocations.top=0,*
```

For information on how to set this parameter, see the reference section on Java System Property Parameters for the UI Servlet in the *Oracle Configurator Installation Guide*.

7

Pricing and ATP

7.1 How it Works

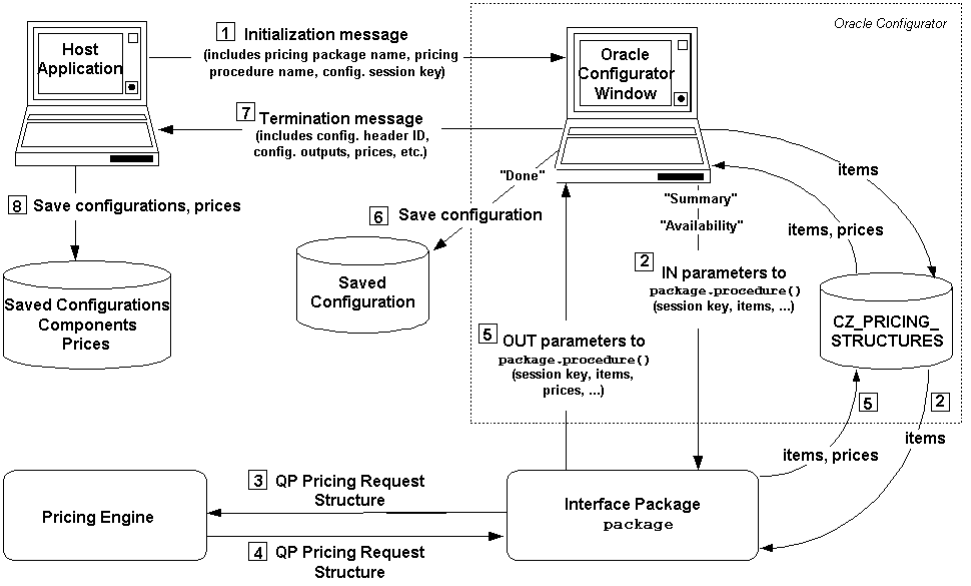
Oracle Configurator allows your host application to display pricing and ATP (Available To Promise) information in the OC configuration window. There is more than one method for presenting this information, to account for these factors:

- The OC configuration window can be called from a variety of different host applications, both Oracle (such as iStore) and non-Oracle.
- Pricing procedures can be accessed from the Oracle Advanced Pricing engine (QP), which has a highly configurable interface.

To accommodate this flexibility, there is no base pricing inherent in Oracle Configurator, and Oracle Configurator does not call the host application's pricing and ATP procedures directly. Instead, Oracle Configurator defines PL/SQL interfaces for pricing and ATP procedures. The host application is responsible for implementing the "callback" pricing and ATP procedures that implement these interfaces. Parameters that identify these procedures are passed to the OC configuration window at initialization; the OC Servlet then calls back to the PL/SQL procedures.

Figure 7–1 shows the architecture for the pricing mechanism in Oracle Configurator. (The architecture for ATP is analogous.)

Figure 7–1 Pricing Architecture in Oracle Configurator



Note: For a fuller description of the pricing architecture for Oracle Configurator, consult the *Oracle Configurator Implementation Guide*.

7.2 What You Do

Integrating the configuration window with your pricing or ATP implementation consists primarily of causing your host application (e.g., through the coding of the “Configure” button) to post the XML initialization message to the OC Servlet, passing as initialization parameters the names of your packages and procedures.

To use the OC pricing and ATP interfaces, you must:

1. Install the OC interface packages in your database, by installing Oracle Configurator with Oracle Rapid Install. See Section 7.2.1, "Database Compatibility" on page 7-4.
2. Write your own PL/SQL pricing or ATP procedures, using the OC interfaces. See the *Oracle Configurator Implementation Guide* for details.

See Section B, "Examples of Pricing and ATP Callback Procedures" on page B-1 for examples.

3. Install your packages containing your procedures into the Oracle Applications database.

   You can interface to the Oracle QP pricing engine from your own procedures.

4. In the initialization message that your host application passes to the OC OC Servlet, provide parameters that specify the name of the pricing package, the procedure to use, and the type of pricing to perform.

   See Section 7.2.4, "Initialization Parameters" on page 7-12 for an example. See Section 3.6.5, "Pricing Parameters" on page 3-12 and Section 3.6.6, "ATP Parameters" on page 3-13 for explanation of the parameters.

7.2.1 Database Compatibility

Oracle Configurator works natively with Oracle Applications Release 11*i*, using the Oracle8*i* Enterprise Edition (Release 8.1.x) database.

In order to obtain pricing data from an Oracle8 Enterprise Edition (Release 8.0.x) database, as used with Oracle Applications 10.7 / 11.0, you must run a concurrent process. See the *Oracle Configurator Implementation Guide.*

There are several likely scenarios for pricing and ATP integration.

These are described in the following table:

To Integrate with...	You would...
Oracle Applications Release 11*i* database	You write your own callback procedures (which can call the QP Advanced Pricing engine). To import BOM data to the Oracle Configurator schema tables, you run concurrent programs in the Oracle Bills Of Material application. To export orders to Order Management (Oracle Applications Release 11*i*), you use existing or new programming in your host application.
Third-party database	For both import and export of pricing data, you must write custom programs.

You can use the callback interface in all these scenarios.

7.2.2 The Pricing Callback Interface

The pricing callback interface package provides interfaces for two distinct procedures:

- Price Single Item
- Price Multiple Items
- Price Multiple Items for MLS

The Price Single Item procedure returns price information for a single item. Table 7–1 describes the parameters for the Price Single Item procedure.

Table 7–1 Price Single Item Procedure Parameters

Parameter	In/Out	Type	Required	Note
`configurator_session_key`	In	Varchar2	Required	Limit of 50 characters
`price_type`	In	Varchar2	Required	Values are: `LIST` or `SELLING`
`ps_node_id`	In	Number	Conditionally Required	Either `ps_node_id` or `item_key` is required.
`item_key`	In	Varchar2	Conditionally Required	Either `ps_node_id` or `item_key` is required.[1]
`quantity`	In	Number	Required	
`uom_code`	In	Varchar2	Required	
`list_price`	Out	Number	n/a	
`selling_price`	Out	Number	n/a	This is the unit selling price. OC automatically multiplies this price by `quantity` to produce the total price (extension) for the item.
`msg_data`	Out	Varchar2	n/a	

[1] If the model was imported from an Oracle BOM, then this key is COMPONENT_CODE:EXPLOSION_TYPE:ORGANIZATION_ID:TOP_ITEM_ID.

The Price Multiple Items procedure returns price information for a group of items. Table 7–2 describes the parameters for the Price Multiple Items procedure.

Table 7–2 Price Multiple Items Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_ session_key	In	Varchar2	Required	
price_type	In	Varchar2	Required	Values are: LIST or SELLING
config_total_price	Out	Number	n/a	

The Price Multiple Items MLS procedure returns price information for a group of items. Table 7–3 describes the parameters for the Price Multiple Items MLS procedure.

Table 7–3 Price Multiple Items MLS Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Varchar2(50)	Required	
price_type	In	Varchar2	Required	Values are: LIST, SELLING or BOTH
config_total_price	Out	Number	Required	
currency_code	Out	Varchar2	Required	
thousands_separator	Out	Varchar2	Required	
decimal_separator	Out	Varchar2	Required	
positive_currency_format	Out	Varchar2	Required	
negative_currency_format	Out	Varchar2	Required	
precision	Out	Varchar2	Required	

The parameters of the interface are passed by positional notation, so you can name the parameters as desired, as long as you retain the positionality specified in Table 7–1, Table 7–2, and Table 7–3.

7.2.2.1 Use of the Database in the Price Multiple Items Procedures

When you specify the Price Multiple Items procedures, Oracle Configurator stores the list of items to be priced in the database table CZ_PRICING_STRUCTURES.

This table is described in Table 7–4 on page 7-7. (See the *Oracle Configurator Technical Reference Manual* for details on Oracle Configurator tables.)

Table 7–4 CZ_PRICING_STRUCTURES Database Table

Column Name	Data Type	Null?	Description
CONFIGURATOR_SESSION_KEY	Varchar2	Not Null	Limit of 50 characters. Primary key. Identifies a configurator session. Only one configuration can be handled in the session.
SEQ_NBR	Number	Not Null	Primary key. Sequence number of the item in the list of items.
PS_NODE_ID	Number		Limit of 9 digits. PS_NODE_ID is a foreign key reference into the CZ_PS_NODES table, which defines the "configuration" identity of the object.
ITEM_KEY	Varchar2	Not Null	Limit of 2000 characters. ORIG_SYS_REF for imported items or PS_NODE_ID for non-imported items.
ITEM_KEY_TYPE	Number	Not Null	Limit of 9 digits. Set to 1[1] if ITEM_KEY is ORIG_SYS_REF. Set to 2[2] if ITEM_KEY is PS_NODE_ID.
QUANTITY	Number		Limit of 9 digits. Item quantity
UOM_CODE	Varchar2		Limit of 3 characters. UOM code
LIST_PRICE	Number		List price
SELLING_PRICE	Number		Selling price
MSG_DATA	Varchar2		Limit of 2000 characters. Message text filled in by your host application.

[1] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_BOM_NODE.
[2] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_PS_NODE.

Your pricing package must retrieve the items from this table and call the pricing engine, then capture all of the results and update the CZ_PRICING_STRUCTURES table with list and / or selling prices. Oracle Configurator retrieves the prices from

the CZ_PRICING_STRUCTURES table during the configuration session, so that they can be presented in the configuration window. When the configuration window exits, OC deletes the pricing records from the CZ_PRICING_ STRUCTURES table.

If your host application needs to retain the prices for use after the end of the current configuration session, then your pricing package must store the results in application-specific tables (or some other location that is insensitive to the Oracle session). Oracle Configurator does not reference the contents of these application-specific tables.

7.2.2.2 Examples of the Pricing Callback Interface

Example 7–1 on page 7-9 shows a possible implementation of the callback interface for both single and multiple-item pricing procedures.

Example 7–3 on page 7-12 shows how you would specify pricing parameters in your initialization message.

Example B–1 on page B-1 provides a minimal example of the use of the callback interface.

Example 7–1 Pricing Callback Interface

```
PACKAGE CZ_PRICE_TEST AUTHID CURRENT_USER AS

 PROCEDURE price_single_item (configurator_session_key IN VARCHAR2,
                              price_type IN VARCHAR2,
                              ps_node_id IN NUMBER,
                              item_key IN VARCHAR2,
                              quantity IN NUMBER,
                              uom_code IN VARCHAR2,
                              list_price OUT NUMBER,
                              selling_price OUT NUMBER,
                              msg_data OUT VARCHAR2);

  PROCEDURE price_multiple_items (p_configurator_session_key IN VARCHAR2,
                                  p_price_type IN VARCHAR2,
                                  p_total_price OUT NUMBER);

END;
```

7.2.3 The ATP Callback Interface

The "Get ATP Dates" procedure returns availability dates for all selected options and for the configuration as a whole. Table 7–5 describes the parameters for the Get ATP Dates procedure.

Table 7–5 ATP Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Varchar2	Required	Limit of 50 characters
warehouse_id	In	Number	Required	
ship_to_org_id	In	Number	Conditionally Required	You must provide either `ship_to_org_id` (by itself), or both `customer_id` and `customer_site_id`.
customer_id	In	Number	Conditionally Required	
customer_site_id	In	Number	Conditionally Required	
requested_date	In	Date	Required	Defaults to value of SYSDATE.
ship_to_group_date	Out	Date	n/a	

The parameters of the interface are passed by positional notation, so you can name the parameters as desired, as long as you retain the positionality specified in Table 7–5.

7.2.3.1 Use of the Database with the ATP Callback Interface

When you specify the Get ATP Dates procedure, Oracle Configurator stores the list of items to obtain ATP dates for in the database table CZ_ATP_REQUESTS. This table is described in Table 7–6 on page 7-10. (See the *Oracle Configurator Technical Reference Manual* for details on Oracle Configurator tables.)

Table 7–6 CZ_ATP_REQUESTS Database Table

Column Name	Data Type	Null?	Description
ATP_REQUEST_ID	Number	Not Null	Primary key.
CONFIGURATOR_SESSION_KEY	Varchar2	Not Null	Limit of 50 characters. Identifies a configurator session. Only one configuration can be handled in the session.

Table 7–6 (Cont.) CZ_ATP_REQUESTS Database Table

Column Name	Data Type	Null?	Description
SEQ_NO	Number	Not Null	Sequence number of the item in the list of items
PS_NODE_ID	Number		Limit of 9 digits. PS_NODE_ID is a foreign key reference into the CZ_PS_NODES table, which defines the "configuration" identity of the object.
ITEM_KEY	Varchar2	Not Null	Limit of 2000 characters. ORIG_SYS_REF for imported items or PS_NODE_ID for non-imported items.
ITEM_KEY_TYPE	Number		Limit of 9 digits. Set to 1[1] if ITEM_KEY is ORIG_SYS_REF. Set to 2[2] if ITEM_KEY is PS_NODE_ID.
QUANTITY	Number		Limit of 9 digits. Item quantity
UOM_CODE	Varchar2		Limit of 3 characters. UOM code
MSG_DATA	Varchar2		Limit of 2000 characters. Message text filled in by your host application.
INV_ORG_ID	Number		Limit of 9 digits.
SHIP_TO_DATE	Date		

[1] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_BOM_NODE.
[2] Value of CZ_PRC_CALLBACK_UTIL.G_ITEM_KEY_PS_NODE.

Your ATP package must retrieve the items from this table and call the `call_atp()` procedure defined in your ATP package, then capture all of the results and update the CZ_ATP_REQUESTS table with ATP dates.

Oracle Configurator retrieves the ATP dates from the CZ_ATP_REQUESTS table during the configuration session, so that they can be presented in the configuration window. When the configuration window exits, OC deletes the ATP dates from the CZ_ATP_REQUESTS table.

If your host application needs to retain the ATP dates for use after the end of the current configuration session, then your ATP package must store the results in application-specific tables (or some other location that is insensitive to the Oracle session). Oracle Configurator does not reference the contents of these application-specific tables.

7.2.3.2 Examples of the ATP Callback Interface

Example 7–2 on page 7-12 shows an implementation of the callback interface for ATP procedures.

Example 7–3 on page 7-12 shows how you would specify ATP parameters in your initialization message.

Example B–3, "Example of Callback ATP Procedure" on page B-2 provides an example in context.

Example 7–2 ATP Callback Interface

```
PACKAGE cz_atp_callback AS

  PROCEDURE call_atp (p_config_session_key IN VARCHAR2,
                      p_warehouse_id IN NUMBER,
                      p_ship_to_org_id IN NUMBER,
                      p_customer_id IN NUMBER,
                      p_customer_site_id IN NUMBER,
                      p_requested_date IN DATE,
                      p_ship_to_group_date OUT DATE);

END cz_atp_callback;
```

7.2.4 Initialization Parameters

Example 7–3 is a test page that shows how you would specify pricing and ATP parameters in your initialization message. The names of the pricing and ATP parameters are typographically emphasized. This example shows parameters for use with Oracle Applications Release 11*i*. See Section 3.6.5, "Pricing Parameters" on page 3-12 and Section 3.6.6, "ATP Parameters" on page 3-13.

Example 7–3 Initialization message using 11i pricing and ATP parameters

```
<html>
<head>
<title>Project Test</title>
<script language="javascript">
```

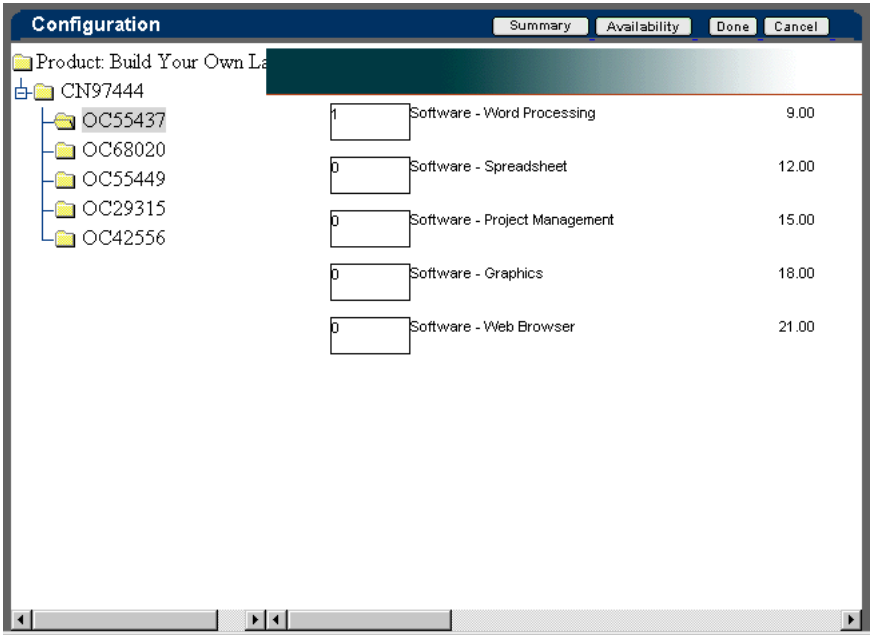
```
function init() {
  document.test1.submit();
}
</script>
</head>

<body onload="init();">
<form
action="http://www.mysite.com:10130/servlets/oracle.apps.cz.servlet.UiServlet"
method="post" id="test1" name="test1"><input type="hidden" name="XMLmsg"
value='<initialize>
<param name="alt_database_name">jdbc:oracle:thin:@server01:1521:vis11</param>
<param name="user">apps</param>
<param name="pwd">apps</param>
<param name="ui_type">DHTML</param>
<param name="gwyuid">APPLSYSPUB/PUB</param>
<param name="fndnam">APPS</param>

<param name="ui_def_id">3080</param>

<param name="pricing_package_name">cz_price_test</param>
<param name="price_mult_items_proc">price_multiple_items</param>
<param name="price_single_item_proc">price_single_item</param>

<param name="configurator_session_key">1234</param>

<param name="atp_package_name">cz_atp_callback_stub</param>
<param name="get_atp_dates_proc">call_atp</param>
<param name="warehouse_id">207</param>
<param name="customer_id">1000</param>

</initialize>'></form>

<br>Loading configurator...

</body>
</html>
```

In order to obtain the final prices calculated by your pricing package, you need to specify a value of `full` for the initialization parameter terminate_msg_behavior. When your configuration session terminates normally, Oracle Configurator returns the final prices in the termination message. Your host application can then save the prices as needed.

7.2.5 Testing Pricing and ATP Integration

Figure 7–2 on page 7-14 shows the effect of the initialization message shown in Example 7–3 on page 7-12.

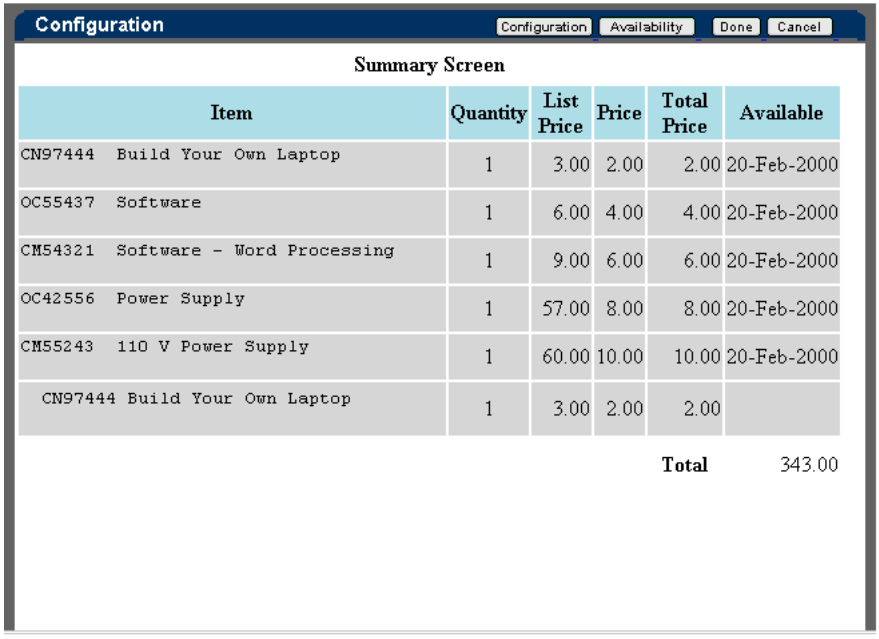
Figure 7–2 Pricing Data in the Configuration Window



In the content pane of the OC configuration window, list prices are displayed next to each item. These are obtained from the database by Oracle Configurator.

Figure 7–3 on page 7-15 shows how pricing and ATP data is displayed when you click the Summary button.

Figure 7–3 Pricing and ATP Data in the Summary Pane



Your pricing procedures are used to calculate the selling prices and total price. If you are using the single-item pricing procedure (specified by the price_single_item_proc parameter), then the total price is calculated by summing up all the selling prices. If you are using the multiple-item pricing procedure (specified by the price_mult_items_proc parameter), then the total price is calculated by your callback procedure.

A

The Checkout Servlet

Example A–1 is the complete source code for Checkout.java, which you can use as a template for constructing your own return URL servlet.

The Java servlet shown here obtains the value of the `valid_configuration` element from the configuration outputs element of the termination message and displays it in an HTML frame that takes the place of the configuration window after your user has closed the window and saved the results of the configuration session.

See the following sections for background.

Section 4.8, "The Return URL"	on page 4-9
Section 4, "Session Termination"	on page 4-1
Section 4.5, "Submission"	on page 4-3
Section 4.5.2, "Configuration Outputs"	on page 4-5
The description of valid_configuration under Section 4.5.1, "Configuration Status"	on page 4-4

The parts of the code that you should customize to work with a different configuration output element than `valid_configuration` are typographically emphasized.

Example A–1 The default Checkout servlet (Checkout.java)

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

import oracle.apps.cz.common.XmlUtil;
import oracle.xml.parser.v2.XMLDocument;
```

```
import org.xml.sax.SAXException;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class Checkout extends HttpServlet {

  // Responds to the UiServlet request containing the <terminate> XML message
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String terminateString = request.getParameter("XMLmsg");
    XMLDocument terminateDoc;
    try {
      terminateDoc = XmlUtil.parseXmlString(terminateString);
    } catch (SAXException se) {
      throw new ServletException(se.getMessage());
    }
    String validConfig = getValidConfig(terminateDoc);
    System.err.println("configuration valid?: " + validConfig);

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<script language=\"javascript\">");
    out.println("parent.location = \"/configurator/Checkout?ValidConfig=" + validConfig + "\"");
    out.println("</script>");
    out.println("</html>");
  }

  // Responds to the secondary request for the page to replace the content frame
  // containing the ValidConfig
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String validConfig = request.getParameter("ValidConfig");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Checked Out with Valid Configuration</title></head>");
    out.println("<body>");
    out.println("Configuration Valid?: " + validConfig);
    out.println("</body>");
    out.println("</html>");
  }

  String getValidConfig(XMLDocument doc) {
    return getTagValue(doc, "valid_configuration", null); // get element from termination msg
```

```
  }

  String getTagValue(XMLDocument doc, String tagName, String defaultValue) {
    Node n = doc.getDocumentElement();
    if (n != null) {
      NodeList nl = n.getChildNodes();
      if (nl != null) {
        for (int i = 0; i < nl.getLength(); i++) {
          Node cn = nl.item(i);
          if (cn.getNodeName().equals(tagName)) {
            NodeList cnl = cn.getChildNodes();
            if (cnl != null) {
              return cnl.item(0).getNodeValue();
            }
          }
        }
      }
    }
    return defaultValue;
  }
}
```

B

Examples of Pricing and ATP Callback Procedures

This appendix contains minimal examples of PL/SQL procedures you might write to use the OC callback interface for pricing and ATP procedures.

See the following sections for background:

Section 7, "Pricing and ATP" on page 7-1

Section 7.2.2, "The Pricing Callback Interface" on page 7-5

Section 7.2.3, "The ATP Callback Interface" on page 7-9

Section 3.6.5, "Pricing Parameters" on page 3-12

Section 3.6.6, "ATP Parameters" on page 3-13

Example B–1 Example of Single-item Callback Pricing Procedure

```
PROCEDURE price_single_item (configurator_session_key IN VARCHAR2,
                             price_type IN VARCHAR2,
                             ps_node_id IN NUMBER,
                             item_key IN VARCHAR2,
                             quantity IN NUMBER,
                             uom_code IN VARCHAR2,
                             list_price OUT NUMBER,
                             selling_price OUT NUMBER,
                             msg_data OUT VARCHAR2) AS
BEGIN

  IF item_key IS NULL THEN
  -- hard-coded price amounts
    list_price := 2.0;
```

```
    selling_price := 2.0;
  ELSE
    list_price := 1.0;
    selling_price := 1.0;
  END IF;

  -- debugging information
  msg_data := configurator_session_key || ' : ' || price_type || ' : ' ||
              item_key || ' : ' || uom_code;

END price_single_item;
```

Example B–2 Example of Multiple-item Callback Pricing Procedure

```
PROCEDURE price_multiple_items (p_configurator_session_key IN VARCHAR2,
                                p_price_type IN VARCHAR2,
                                p_total_price OUT NUMBER) AS
BEGIN
  update cz_pricing_structures set list_price = 3.0*seq_nbr,
    selling_price = 2.0*seq_nbr,
    where configurator_session_key =
    p_configurator_session_key;

-- calculation using pricing table for storage
    select sum(selling_price) into p_total_price from cz_pricing_structures
    where configurator_session_key = p_configurator_session_key;

  -- hard-coded price amount
  -- p_total_price := 343.00;

END price_multiple_items;
```

Example B–3 Example of Callback ATP Procedure

```
  PROCEDURE call_atp (p_config_session_key IN VARCHAR2,
                      p_warehouse_id IN NUMBER,
                      p_ship_to_org_id IN NUMBER,
                      p_customer_id IN NUMBER,
                      p_customer_site_id IN NUMBER,
                      p_requested_date IN DATE,
                      p_ship_to_group_date OUT DATE) IS
  BEGIN
```

```
  update cz_atp_requests set ship_to_date = sysdate-10
    where configurator_session_key
    = p_config_session_key;

  p_ship_to_group_date := sysdate;
END call_atp;
```

Glossary of Terms

This glossary for Oracle Configurator is followed by a Glossary of Acronyms

Active Model

The part of Oracle Configurator runtime architecture that processes model structure and rules to create configurations. Interfaces dynamically with the end user Active UI and data.

Active User Interface

The part of Oracle Configurator runtime architecture that provides the graphical views necessary to create configurations interactively. Interfaces with the Active Model and data to give users access to customer requirements gathering, product selection, and customer-centric extensions.

Applet

A Java application running inside a web browser. See also Java and Servlet.

Application Architecture

The software structure of an application at runtime. Architecture affects how an application is used, maintained, extended, and changed.

Architecture

The software structure of a system. Architecture affects how a system is used, maintained, extended, and changed. See also Application Architecture.

Beta

An external release, delivered as an installable application, and subject to system, validation, and acceptance testing. Specially selected and prepared end users may participate in beta testing.

Bill of Materials (BOM)

A list of component items associated with a parent item (assembly) and information about how each item relates to the parent item.

BOM

See Bill of Material.

BOM Item

The nodes imported into the Oracle Configurator Developer Model that correspond to an Oracle BOM.

BOM Model

A model that you import from Oracle Bills of Material (BOM) into Oracle Configurator Developer. When you import a BOM Model, the effective dates defined for each component in Oracle BOM are also imported into Configurator Developer. You can extend the Structure of the BOM Model in Configurator Developer, but you cannot modify the BOM Model itself or any of its attibutes.

BOM Model Node

The imported Model node in the Oracle Configurator Developer that corresponds to Standard Model in an Oracle Bill of Materials.

BOM OptionClass

The imported Model node in the Oracle Configurator Developer that corresponds to Option Class in an Oracle BOM.

BOM StandardItem

The imported Model node in the Oracle Configurator Developer that corresponds to Standard Item in an Oracle BOM.

Boolean Expression

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

Bug

See Defect.

Build

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

CIO

See Oracle Configuration Interface Object.

Client

A runtime program using a server to access functionality shared with other clients.

Comparison Rule

An Oracle Configurator Developer rule type to establish a relationship that determines 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 to establish a relationship among features in the Model that specifies the allowable combinations of options. See also, Property-based Compatibility Rule.

Compatibility Table

A type of compatibility relationship where the allowable combination of options are explicitly enumerated.

Component

Represents a configurable element in a product. An element of the Model structure, typically containing features. May correspond to one screen of selections in an Oracle runtime configurator.

Component Set

An element of the Model that contains a number of components of the same type, where each component of the set is independently configured.

Concurrent Manager

A process manager that coordinates the concurrent processes generated by users' concurrent requests. An Oracle Applications product group can have several concurrent managers.

Concurrent Process

A task that can be scheduled and is run by a concurrent manager. A concurrent process runs simultaneously with interactive functions and other concurrent processes.

Concurrent Processing Facility

An Oracle Applications facility that runs time-consuming, non-interactive tasks in the background.

Concurrent Program

Executable code (usually written in SQL*Plus or Pro*C) that performs the function(s) of a requested task.

Concurrent Request

A user-initiated request issued to the concurrent processing facility to submit a non-interactive task, such as running a report.

Configuration

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

Configuration Model

The model structure and rules-based content of an Oracle runtime configurator. The configuration model is constructed and maintained using Oracle Configurator Developer, and is interpreted at runtime by the Active Model.

Configuration Rules

The Oracle Configurator Developer logic rules and numeric rules available for defining configurations.

Configurator

The part of an application that provides custom configuration capabilities.

Constraint Rule

An Oracle Configurator Developer rule type to create a logical relationship among features and options. See also Rules.

Contributes to

An Oracle Configurator Developer numeric rule type for accumulating a total value.

Consumes from

An Oracle Configurator Developer numeric rule type for specifying the quantity of a resource used.

CRM

Customer Relationship Management. The aspect of the enterprise that involves contact with customers, from lead generation to support services.

Customer

The person or persons for whom products are configured by end users of the Oracle Configurator or other ERP and CRM applications.

Customer-centric Extensions

See Customer-centric Views.

Customer-centric Views

Optional extensions to core functionality that supplement product selection with rules for pre-selection, validation, and intelligent views. View capabilities include generative geometry, drawings, sketches and schematics, charts, performance analyses, and ROI calculations.

Customer Requirements

The needs of the customer that serve as the basis for determining the configuration of products, systems, and/or services. Also called Needs Assessment.

Data Import

Populating the Oracle Configurator schema with enterprise data from ERP or legacy systems via import tables.

Data Integration Object

Also known as the DIO, the Data Integration Object is a server in the runtime application that creates and manages the interface between the client (usually a user interface like the Active User Interface) and the Oracle Configurator schema.

Data Maintenance Environment

The environment in which the Oracle runtime configurator data is maintained.

Data Replication

The activity of downloading and uploading configuration, quote, and order data between the Oracle Configurator schema on the enterprise server and Oracle Configurator Mobile Database on end-user mobile laptop PCs. See also Data Synchronization.

Datasource

A programmatic reference to a database. Referred to by a datasource name, or DSN.

Data Synchronization

A process for matching the data in the Oracle Configurator schema and the data available to client processes such as the Oracle SellingPoint application. See also Data Replication.

Default

The automatic selection of an option based on the pre-selection rules or the selection of another option.

Defaults

An Oracle Configurator Developer logic rule to determine the logic state of features or options in a default relation to other features and options. For instance, if you set A to True by selecting it, B becomes true (selected) if it is available (not false) and can be set to True without contradicting a non-default rule or a previous default setting for B.

Defect

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

Defect Tracking

A system of identifying defects for managing additional tests, testing, and approval for release to users.

Deliverable

A work product that is specified for review and delivery.

Demonstration

A presentation of the tested application, showing a particular usage scenario.

Design Chart

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

Design Review

A technical review that focuses on application or system design.

Developer

The tool (Oracle Configurator Developer) used to create configuration models. The person who uses Oracle Configurator Developer to create a configurator. See also Implementer.

DIO

See Data Integration Object.

End User

The ultimate user of the Oracle runtime 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 web or kiosk.

Enterprise

The systems and resources of a business.

Environment

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

ERP

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

Excludes

An Oracle Configurator Developer rule type for determining the logic state of features or options in an excluding relation to other features and options. For instance, if you set A to True, B becomes false, since it is not allowed when A is true. If you set A to False, there is no effect on B, meaning it could be true, false, or unknown.

Extended Functionality

A release after delivery of core functionality that extends that core functionality with customer-centric views, more complex proposal generation, discounting, quoting, and expanded integration with ERP, CRM, and third-party software.

Feature

An element of the Model structure. A configurable parameter of a component. Features can either have a value (numeric or boolean) or enumerated options.

Functional Companion

An object associated with a component that supplies methods that can be used to initialize, validate and generate customer-centric views and outputs for the configuration.

Functional Specification

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

Incremental Construction

The process of organizing the construction of the application into builds, where each build is designed to meet a specified portion of the overall requirements and is unit tested.

Implementation

The stage in a project between defining the problem by selecting a configuration technology vendor, such as Oracle, and deploying the completed sales configuration application. The Implementation stage includes gathering requirements, defining test cases, designing the application, constructing and testing the application, and delivering it to users.

Implementer

The person who uses Oracle Configurator Developer to build the model structure, rules, and UI customizations that make up an Oracle runtime configurator.

Implies

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 instance, if you set A to True by selecting it, B becomes true, since selecting A implies that B is also selected. If you set A to False by deselecting it, there is no effect on B, meaning it could be true false or unknown based on other relations B participates in. And if you set B to True by selecting it, there is no effect on A, meaning it could be true false or unknown based on other relations A participates in. But if you set B to False by deselecting it, the relation of A implies B is preserved only by having A be false (deselected) as well.

Import Tables

Tables mirroring the Oracle Configurator schema Item Master structure, but without integrity constraints. Import Tables allow batch population of the Oracle Configurator schema Item Master. Import Tables are used in conjunction with extractions from Oracle Applications or legacy data to create, update, or delete records in the Oracle Configurator schema Item Master.

Install

A program that sets up the local machine and installs the application for testing and use.

Instantiate

To create an instance of a configuration model Component, Feature, or Option in the User Interface.

Integration

The process of combining multiple software components and making them work together.

Integration Testing

Testing the interaction among software programs that have been integrated into an application or system.

Intelligent Views

Configuration output, such as reports, graphs, schematics, and diagrams, that help to illustrate the value proposition of what is being sold.

Item Master

A table in the Oracle Configurator schema containing data used to structure the product. Data in the item master is either entered manually or imported from Oracle Applications or legacy data.

Item Type

A table in the Oracle Configurator schema containing data used to classify the product data in the item master table.

Java

An object-oriented programming language commonly used in internet applications, where Java applications run inside web browsers and servers. See also Applet and Servlet.

Log File

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

Logic Rules

Logic rules directly or indirectly set the logical state (true, false, or unknown) of features and options in the Model.

There are four (4) primary logic rules: Implies, Requires, Excludes, and Negates. Each of these rules takes a list of features or options as operands. See also Logic, Implies, Requires, Excludes, and Negates.

Maintainability

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

Maintenance

The effort of keeping a system running once it has been deployed, through bug fixes, procedure changes, infrastructure adjustments, data replication schedules, etc.

Maintenance Guide

A guide for maintaining a specific application or system. The maintenance guide covers all aspects of maintenance described in the generic Maintenance Plan.

Maintenance Plan

A document that outlines what is required for successful maintenance, and who is responsible for all the actions and deliverables of carrying out maintenance on a system.

MDUI

See Model-driven UI.

Mobile Database

See Oracle Configurator Mobile Database.

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. May consist of BOM Items.

Model-driven UI

The graphical views of the model structure and rules generated by the Active UI to present end users with interactive product selection based on configuration models.

Model Structure

Hierarchical, “tree” view of data in terms of product elements (Models, Products Components, Features, Options, BOM Models, BOM OptionClasses, BOM StandardItems, Resources, and Totals). May include reusable components.

MRP

Manufacturing Resource Planning. A software system and process for monitoring and maintaining the customer's manufacturing systems.

Negates

An 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 instance, if you set one item in the relationship to True, the other item must be false. And if you set one item to False, the other item must be true.

Node

The place in a Model occupied by a component, feature, option or variable, BOM Model, BOM OptionClass, or BOM StandardItem.

Numeric Rules

Rules that are used to set the global parameters specified in product structuring. See also, Contributes to and Consumes from.

OC

See Oracle Configurator.

Opportunity

The workspace in the Oracle SellingPoint application and Oracle Sales Online in which products, systems, and / or services are configured, quotes and proposals are generated, and orders are submitted.

Option

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

A logical selection made by the end user when configuring a component.

Oracle Configurator

The product family consisting of development tools and runtime applications such as Oracle Configurator schema, Oracle Configurator Developer, run-time Oracle Configurator, and Oracle SellingPoint application. Also the Oracle runtime configurator variously packaged for use in networked, mobile, or web deployments.

Oracle Configurator Architecture

The application runtime architecture consists of the Active User Interface, the Active Model, and the Oracle Configurator schema or Oracle Configurator Mobile Database. The application development architecture consists of Oracle Configurator Developer and the Oracle Configurator schema, with test instances of an Oracle runtime configurator.

Oracle Configurator Developer

The suite of tools in the Oracle Configurator product family for constructing and maintaining configurators.

Oracle Configuration Interface Object (CIO)

A server in the runtime application that creates and manages the interface between the client (usually a user interface like the Active User Interface) and the underlying representation of model structure and rules in the Active Model.

CIO protocols support creating and navigating the Model, querying and modifying selection states, and saving and restoring configurations.

Oracle Configurator Mobile Database

The runtime version of the standard Oracle Configurator schema that manages data for the configuration model in a mobile deployment. The runtime schema includes customer, product, and pricing data as well as data created during operation of an Oracle Configurator.

Oracle Configurator Schema

The implementation version of the standard Oracle runtime 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.

Oracle SellingPoint Application

The test application generated by Oracle Configurator Developer. Also a full configuration environment with opportunity management, quotes, and proposals for networked or mobile deployments.

Output

The output generated by a configurator, such as quotes, proposals, bills of material (BOM), and customer-centric views.

PDM

Product Data Management. A software system that manages the version control of product data.

Populator

An entity in the Oracle Configurator Developer that defines how to create a Model from information in the item master.

Pre-selection

The default state in a configurator that defines an initial selection of components, features, and options for configuration.

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

Principal Design Consultant

Member of the project team responsible for architecting the design of the application.

Product

Whatever is subjected to configuration and is the output of the application.

The root element of the Model.

Product Family

A collection of products or product lines, which are organized as a group by a provider or manufacturer.

Project

The workspace in Oracle Configurator Developer in which configurators are constructed

Project Manager

A member of the project team who is responsible for directing the project during implementation.

Project Plan

A document that outlines the logistics of successfully implementing the project, including the schedule.

Property

A named value associated with an object in the Model or the item master. A set of properties may be associated with an item type.

Property-based Compatibility Rule

A kind of compatibility relationship where the allowable combinations of options are specified implicitly by relationships among property values of the options.

Prototype

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

Reference

The use of a reusable component within the Model. Not implemented in Release 11*i* or before.

Regression Test

An automated test that ensures the newest build still meets previously tested requirements and functionality.

Requires

An Oracle Configurator Developer logic rule type that determines the logic state of features or options in a requirement relation to other features and options. For instance, if you set one item in the relationship to True, the other item is required to be true as well. And if you set one item to False, the other item must be false as well.

Resource

Staff or materials available or needed within an enterprise.

A variable in the Model used to maintain the balance of features not consuming more of a specific resource than has been provided by other features.

Reusable Component

A component that is referenced from multiple locations in the Model. Not implemented in Release 11*i* or before.

Reusability

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

Rules

Also called business rules or configuration rules. 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 are used to provide pre-selection and validation capability in an application.

See also Logic Rules and Numeric Rules.

Runtime

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

Sales Configuration

A part of the sales process to which configuration technology has been applied in order to increase sales effectiveness and decrease order errors. Commonly identifies needs assessment and product configuration.

Server

Centrally located software processes or hardware, shared by clients.

Servlet

A Java application running inside a web server. See also Java and Applet.

Solution

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

System

The hardware and software components and infrastructure integrated to satisfy functional and performance requirements.

Test Case

A description of inputs, execution instructions, and expected results, which are created for the purpose of determining whether a specific software feature works correctly or a specific requirement has been met.

Total

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

Undetermined

The logic state that is neither true nor false, but unknown at the time a logic rule is executed. This logic state is also referred to as Available, especially when considered from the point of view of the Oracle runtime configurator end user.

Unit Test

Execution of individual routines and modules by the application implementer or by an independent test consultant for the purposes of finding defects.

Update

Moving a production configurator to a new version of configuration model.

Upgrade

Moving the configurator to a new release of Oracle Configurator.

User

The person using the Oracle Configurator Developer tools and methods to build an Oracle runtime configurator. See also 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.

User Requirements

A description of what the Oracle Configurator or Oracle SellingPoint application is expected to do from the end user's perspective.

User's Guide

Documentation on using the application or configurator to solve the intended problem.

Validation

Tests that ensure that the configured components will meet specific performance or acceptance criteria.

A type of functional companion that is implemented to ensure that the configured components will meet specific performance or acceptance criteria.

Variable

Parts of the Model that are represented by Totals, Resources, or numeric Features.

Verification

Tests that check whether the result agrees with the specification.

Glossary of Acronyms

API

Application Programming Interface

ATP

Available to Promise

BOM

Bill of Material

CIO

Configuration Interface Object

CM

Configuration Management

COM

Component Object Model

CRM

Customer Relationship Management

DBMS

Database Management System

DCOM

Distributed Component Object Modeling

DHTML

Dynamic Hypertext Markup Language

DIO

Data Integration Object

DLL

Dynamically Linked Library

DXF

Drawing Exchange Format (AutoCAD drawings)

ECO

Engineering Change Order

ERM

Enterprise Relationship Management

ERP

Enterprise Resource Planning

ESD

Electronic Software Distribution

ESP

External Service Provider

ESS

Enterprise Selling System

HT

High Tech

HTML

Hypertext Markup Language

IP

Industrial Products

IS
Information Services

ISS
Interactive Selling System

ISV
Independent Software Vendor

LAN
Local Area Network

MAPI
Messaging Application Programming Interface

MC/S
Mobile Client / Server System

MDUI
Model-Driven User Interface

MES
Marketing Encyclopedia System (Catalog)

MIS
Management Information Systems

MRP
Manufacturing Resource Planning

MS
Microsoft

OC
Oracle Configurator

OCX
Object Control File, OLE custom controls

ODBC
Open Database Connectivity

OLE
Object linking and embedding

OMS
Opportunity Management System

OOD
Object-Oriented Design

ORB
Object Request Broker

PDM
Product Data Management

PIA
Project Impact Assessment

POS
Point of Sale

QA
Quality Assurance

RAD
Rapid Application Development

RDBMS
Relational Database Management System

RFQ
Request for Quote

ROI
Return on Investment

SAS
Sales Analysis System

SCM
Supply Chain Management

SCS
Sales Configuration System

SE
Sales Engineer

SFA
Sales Force Automation

SI
System Integrator

SOT
Strategic Options Theory

SQA
Software Quality Assurance

SQL
Structured Query Language

TERM
Technology-Enabled Relationship Management

TES
Technology-Enabled Selling

UI
User Interface

VAR
Value-Added Reseller

VB

Microsoft Visual Basic

WAN

Wide Area Network

WIP

Work In Progress

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z