Oracle® Configurator Toolkit

Oracle Configurator Toolkit Developer's Guide

Release 11*i*

May 2000

Part No. A77067-05

Oracle Configurator Toolkit (OCT) enables you to integrate the Oracle Configurator into your own Internet application. This document describes how to embed the OCT into an Internet host application, such as a custom web store.

Oracle Configurator Toolkit Developer’s Guide, Release 11*i*

Part No. A77067-05

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

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle Configurator Toolkit Developer’s Guide, Release 11*i*

Part No. A77067-05

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

The Oracle Configurator Toolkit (OCT) is a separately purchasable Oracle product that enables you to integrate the Oracle Configurator into your own Internet application.

This manual describes how to embed the OCT into an Internet application, such as a custom web store.

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

Intended Audience

This manual is intended primarily for developers of custom web stores.

Technologies and Tools

Working with the OCT 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 Developer Tutorial*
- *Oracle Configurator and SellingPoint Administration Guide*
- *Oracle Configuration Interface Object (CIO) Developer's Guide*

The following documents may also be useful:

- *Oracle8i JDBC Developer's Guide and Reference*
- *Oracle Application Server 4.0 Developer's Guide: JServlet Applications*

Structure

This manual contains the following chapters and appendices:

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.

1 Introduction

Typical Tasks

This document will help you complete the following tasks, which you must perform in order to integrate the Oracle Configurator Toolkit (OCT) into your internet host application:

	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 Toolkit servlet on your internet application server.	Chapter 2, "Installing the Servlet"
	Tailor the initialization message that invokes the configuration window and the return URL that handles the configuration outputs.	Chapter 3, "Integration: Session Initialization"
	Extract the required data from the XML termination message produced by the configuration window.	Chapter 4, "Integration: Session Termination"
	Provide pricing and ATP data in the configuration window.	Chapter 5, "Integration: Pricing and ATP"
	Customize certain HTML files to modify the appearance of the configuration window.	Chapter 6, "User Interface"
	Test the configuration window in the host application, running in an internet browser.	Chapter 7, "Testing"

Toolkit Architecture

Three-tier Web Application Architectures

The OCT is a toolkit for deploying a web-based configuration window in a three-tier architecture:

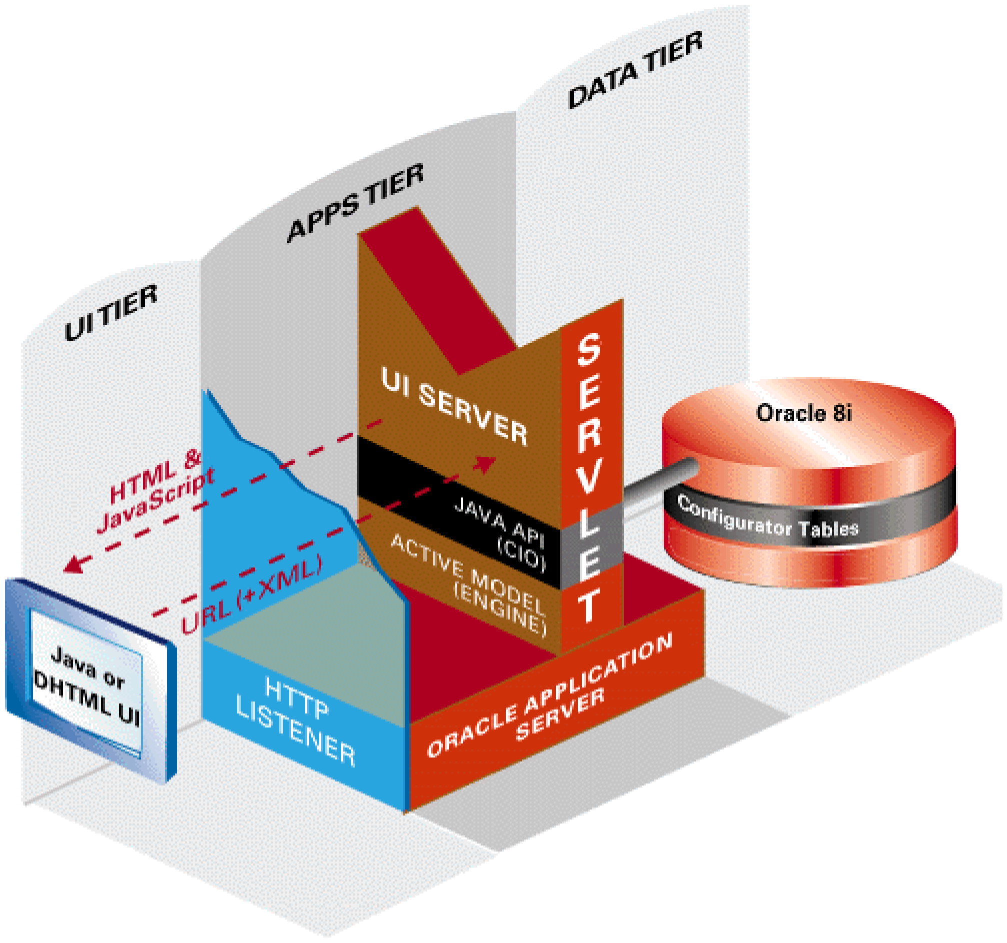
Table 1–1 Three-tier web application architecture

Tier	Tier components	OCT Components
Client	thin client	your host application, running in a browser
		OCT configuration window (DHTML)
Application	application (web) server	Oracle Application Server
	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 (such as Oracle 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 Architectural Overview of Oracle Configurator Toolkit



The UI Servlet is installed on the application server, along with the other server-side components of Toolkit. The database may or may not be on that machine, as well.

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.

Application Flow

1. An initialization message is sent to the UI 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.

   If you need to make changes to your configuration rules, you stop and restart your web server to reflect the rule changes. If you are running multiple web servers, you can redirect user requests to another server running an updated Model.
3. The UI 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 XML message results, which are translated into 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.

Elements Needed and Provided

Table 1–2 lists all the elements needed by a web application using Oracle Configurator Toolkit, 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 Toolkit

Element Needed	Element Provided?	Action Required?	Action
Your host application (e.g., Oracle iStore or other web store)	No	Yes	You implement the application.
Web server (e.g., Oracle Application Server)	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 UI Servlet.
Oracle Configurator UI Servlet	Yes	No	The UI Servlet encapsulates other elements: ▪ the UI Server ▪ the CIO ▪ the logic engine
Configurator UI Server	Yes	No	Automatically accessed by UI Servlet to process user interface
Oracle Configurator CIO	Yes	No	Automatically accessed by UI Servlet. (Also available for custom programming.)
Oracle Configurator logic engine	Yes	No	Automatically accessed by CIO, through UI Servlet.
DHTML configuration window	Yes	No	Configuration window is produced automatically by UI 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.

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

Element Needed	Element Provided?	Action Required?	Action
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.
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 UI 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 UI 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 Installing the Servlet

In order to use Toolkit, you must first install the UI Servlet on your Internet server. This process consists of the following tasks:

Table 2–1 Overview of tasks for installing the servlet

For This Task ...	See ...
Installing the required files.	Installing the Required Files on page 2-2
Moving files to alternate locations. (Optional)	Required Files and Locations on page 2-2
Installing the servlet on your Internet server.	Creating an Application and JServlet Cartridge on page 2-9
Modifying the configuration parameter values of your Internet server.	Configuring an Application and JServlet Cartridge on page 2-12
If you are installing multiple servlets, making certain adjustments. (Conditional)	Multiple Servlet Scenarios on page 2-19

Note: This installation procedure is specific to Oracle Application Server (OAS) on the Windows NT and Solaris platforms. You may need to adapt the procedure for other Internet servers. Oracle Configurator Toolkit requires Oracle Application Server 4.0.8.1, or any Internet server that supports servlets.

Related Documentation

You will need to consult the following documentation when installing the Toolkit servlet in Oracle Application Server:

- Oracle Application Server 4.0.8 Documentation Set

Prerequisites

- You must have installed Oracle Applications Release 11*i*, specifically including Oracle Configurator.
- You must have installed your internet server. Oracle Configurator Toolkit has been tested with Oracle Application Server 4.0.8.1.
- You must have JDBC connectivity between OAS and your Oracle database. This may include copying or editing connectivity parameters in the TNSNAMES.ORA file for OAS.
- Check the *Oracle Configurator and SellingPoint Release Notes*, and the chapter on deployment in the *Oracle Configurator and SellingPoint Administration Guide*, for any other requirements.

Installing the Required Files

The required files must be in place before installing the servlet in your Internet server. This is normally done automatically, as described in Table 2–2.

- You may wish to move certain files to other locations, to suit the needs of your site or host application. If so, see Required Files and Locations on page 2-2.
- If you don't need to move any files, skip to Creating an Application and JServlet Cartridge on page 2-9.

Table 2–2 Default Installation Methods for Toolkit

Release	Platform	To install the required files...
Release 11*i*	Windows NT Solaris	On a Windows NT machine, run the setup program on the Oracle Configurator CD (SPSetup.exe), choosing the "Toolkit" option.
Release 11*i*	Solaris	On a Solaris machine, run the Oracle Installer, after having installed Oracle Applications (for instance, with Autoinstall).

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.
- For Release 4.2.1, you may wish to move the files from the Windows NT machine on which they are installed to a Solaris machine.

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.

General Directory Structure

Table 2–3 shows the directories required for Toolkit, 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 2–3 General Structure of Directories for OCT

Directory Role	Description
OCT Installation	This directory is referenced in these instructions as `oct_install`. The directory in which you install OCT, based on your choice of installation directory in the Oracle Configurator setup program. On Windows NT, this defaults to `ORACLE_HOME\OSP`. On Solaris, this defaults to `$APPL_TOP`.
Servlet	Contains the Java class or archive files that implement the UI Servlet.
HTML	Contains the HTML template files that define the configuration window of your host application (see Chapter 6).
Media	Contains the image files used by the configuration window of your host application.
Active Model	Contains the Active Model files created by Configurator Developer, which define the configuration model called by your host application.
Log	Contains log files written by the UI Servlet when the configuration window is used.

Table 2–4 and Table 2–5 provide examples of this general directory structure and naming applied to the Windows NT and Solaris platforms.

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 Toolkit servlet.

The Active Model directory is empty until you begin using Configurator Developer. The Log directory is empty until you begin using the OCT configuration window.

Directory and file names are case-sensitive on Solaris.

Certain HTML files are language-specific, and are stored in language-specific directories, such as `html/US`. See Table 2–8 on page 2-6 for a list of these files.

Table 2–4 OCT Directories for Windows NT

Directory	Example
OCT Install	`D:\orant\OSP\`
Servlet	`D:\orant\OSP\OSP\`
HTML	`D:\orant\OSP\WebUI\html\`
Media	`D:\orant\OSP\WebUI\media`
Active Model	`D:\orant\OSP\Shared\ActiveModel\`
Log	`C:\temp\`

Table 2–5 OCT Directories for Solaris

Directory	Examples
OCT Installation	`/d1/my-oct, $APPL_TOP`
Servlet	`/d1/my-oct/servlet/, $APPL_TOP`
HTML	`/d1/my-oct/html/, $APPL_TOP/html`
Media	`/d1/my-oct/media/, $APPL_TOP/media`
Active Model	`/d1/my-oct/activemodel/`
Log	`/d1/my-oct/log/`

File Types

Table 2–6 shows the types of files included in an installation of Toolkit.

Table 2–6 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 UI Servlet, UI Server, configuration engine, and other elements of Toolkit.
Java Class	.class	Java class files that implement application logic.
Java Source	.java	Java source code files that can be modified to customize the behavior of the UI Servlet with your host application.
JavaScript Control	.js	JavaScript control files that implement the behavior of the configuration window.
Media	.gif, .jpg	Image files used by your host application.
Shared Object	.so, .dll	The platform-specific implementation of the Internet server cartridge.
XSL Stylesheet	.xsl	XSL stylesheets that transform XML output from the UI Server into JavaScript.

Files for the Servlet Directory

Table 2–7 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 *`oct_install`* directory, rather than in a separate `/servlet` subdirectory.

For the physical location of the Servlet directory, and examples by platform, see Table 2–4 on page 2-4 and Table 2–5 on page 2-4.

Table 2–7 Files for the Servlet Directory

File	For Platform	Comment
Java Classes		
apps.zip	Solaris	Includes all the Java classes required for OCT, in addition to those for Oracle Applications.

Table 2–7 (Cont.) Files for the Servlet Directory

File	For Platform	Comment
cz3rdpty.jar	Windows NT	Includes the Java classes for collections, the XML parser, the Swing UI, and various fundamentals.
config.jar	Windows NT	Includes the Java classes in the package oracle.apps.cz.servlet that implement the UI Servlet, notably these: UiServlet, Proxy, and Logout.
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 OCT installation program.
czjni.dll	Windows NT	
libcz.so	Solaris	Must be in the LD_LIBRARY_PATH environment variable parameter for your cartridge. See Environment Variables Parameters on page 2-14.
libczjni.so	Solaris	

Files for the HTML Directory

The files listed in Table 2–8 must be located in the HTML directory, on all platforms.

For the physical location of the HTML directory, and examples by platform, see Table 2–4 on page 2-4 and Table 2–5 on page 2-4.

Table 2–8 Files for the HTML Directory

File
HTML Templates
czCntnt.htm
czdisp.htm
czSource.htm
cztree.htm
Language-Specific HTML Templates
czFraIE.htm
czFraNS.htm
czHeader.htm
czLeft.htm

Table 2–8 (Cont.) Files for the HTML Directory

File
czRight.htm
XSL Stylesheets
czclient.xsl
czcmdcvt.xsl
czserver.xsl
czxml2js.xsl
JavaScript Controls
czProxy.js
czAll.js (for Internet Explorer browsers)
czJSSrc.jar (for Netscape browsers)

Files for the Media Directory

The files listed in Table 2–9 must be located in the Media directory, on all platforms.

Table 2–9 Files for the Media Directory

czAvail0.gif	czFCUnSa.gif	czSIbox1.gif
czAvail1.gif	czfldcls.gif	czSIdn0.gif
czBack0.gif	czfldclu.gif	czSIdn1.gif
czBack1.gif	czfldopn.gif	czSIlt0.gif
czblank.gif	czflopnu.gif	czSIlt1.gif
czblbar.gif	czFOpen.gif	czSIrt0.gif
czbollf.gif	czForw0.gif	czSIrt1.gif
czbollt.gif	czForw1.gif	czSIup0.gif
czboluf.gif	czFOUnSa.gif	czSIup1.gif
czbolun.gif	czlgblf.gif	czsplogo.gif
czbolut.gif	czlgblt.gif	czSrBg.gif
czButC0.gif	czlgbuf.gif	czSumm0.gif
czButC1.gif	czlgbun.gif	czSumm1.gif

Table 2–9 (Cont.) Files for the Media Directory

czButL0.gif	czlgbut.gif	czTabBL.gif
czButL1.gif	czLNode.gif	czTabBR.gif
czButR0.gif	czMLNode.gif	czTabC.gif
czButR1.gif	czMNode.gif	czTabL.gif
czCanc0.gif	czMore.gif	czTabR.gif
czCanc1.gif	czoptlf.gif	czTNode.gif
czCboArr.gif	czoptlt.gif	czUnsat.gif
czcomlf.gif	czoptuf.gif	czWarn.gif
czcomlt.gif	czoptun.gif	czyellgr.gif
czcomuf.gif	czoptut.gif	
czcomun.gif	czPLNode.gif	
czcomut.gif	czPNode.gif	
czConf0.gif	czSHbar0.gif	
czConf1.gif	czSHbar1.gif	
czDone0.gif	czSHbox0.gif	
czDone1.gif	czSHbox1.gif	
czFClose.gif	czSIbox0.gif	

Creating an Application and JServlet Cartridge

This section contains a summary of the steps required for creating an application on Oracle Application Server (OAS), and installing the OCT JServlet cartridge in the application.

- For full details on this task, consult the instructions on cartridge and component administration in the *Oracle Application Server 4.0 Administration Guide*.
- You will need to log into the OAS Manager as the "admin" user in order to perform these steps.
- In this summary, *`app_name`* is the name that you choose for the application that you are adding to OAS Manager. Note that "application" has a specific meaning in OAS, and that this "application" is the OCT UI Servlet, *not* your host application.

  In OAS, a cartridge consists of code that executes application logic and configuration data that enables the cartridge to locate the application logic. A cartridge can provide a runtime environment for a specific programming language. Oracle Configurator Toolkit provides a JServlet cartridge, which contains a Java Virtual Machine for running Java class files.
- In this summary, *`servlet_vpath`* is to be replaced with the value `/servlet/`, for compatibility with certain of the OCT HTML Template files.

Note: You should not have to set a different value for *`servlet_vpath`* unless you are installing multiple OCT servlet cartridges on the same Internet server. If you are, see Multiple Servlet Scenarios on page 2-19.

Table 2–10 Summary Steps for Creating an Application and JServlet Cartridge

	Window/Page/ Frame	**Button/Field**	**Action**	**Because...**
1	OAS Welcome Page	⊞	Click	Displays the components on the site: Oracle Application Server, HTTP Listeners, Applications
2		⊞ display of components	Select the “Applications” node (not the ⊞ next to “Applications”)	Displays the existing applications in the right frame.
3	Applications page in the right frame	➕ (Add)	Click	Opens Add Application dialog.
4	Add Application dialog	Application Type field	Choose `JServlet`	Adds the JServlet type of application to the applications on the site.
5		Configure Mode field	Choose `Manually`	Enables you to enter configuration data using dialog boxes. (The other option, `FromFile`, requires getting the same configuration data from a registration file.)
6		Apply button	Click	Displays a second Add Application dialog
7	Second Add Application dialog	Application Name	Enter *`app_name`*	This is the name used to identify the application in OAS. Note that this application is the OCT UI Servlet, *not* your host application.
8		Display Name	Enter *`app_name`*	Specifies the application name that appears on the navigational OAS Tree Applet. If you desire, this can differ from the Application Name.
9		Application Version	(You can leave this blank)	This is the version of the application you are adding.
10		Apply button	Click	Launches Success dialog box, which contains a button that enables you to add cartridges to the application
11	Success dialog box	Add Cartridge to this Application button	Click	Displays the Add A Cartridge dialog.

Table 2–10 (Cont.) Summary Steps for Creating an Application and JServlet Cartridge

	Window/Page/ Frame	**Button/Field**	**Action**	**Because...**
12	Add A Cartridge dialog	Cartridge Name	Enter *`cart_name`*	Specifies the name used to identify your JServlet cartridge in your *`app_name`* OAS application. This can match the Application Name.
13		Display Name	Enter *`cart_name`*	Specifies the cartridge name that appears on the navigational OAS Tree Applet. If you want, this can differ from the Cartridge Name.

14		Virtual Path	Enter *`servlet_vpath`*
			Which should be:
			`/servlet/`
				Specifies the virtual path relative to the Internet server host name that clients will use to invoke the JServlet cartridge in a URL, in the form:
				`http://`*`hostname`*`/servlet/`*`filename`*`.htm`
				For example:
				`http://www.mysite.com/servlet/test.ht m`
				■ Note: Enter the virtual path exactly as indicated, since this path is incorporated into certain of the OCT HTML Template files. Entering a different virtual path requires you to modify these files. You should not have to set a different value for *`servlet_vpath`* unless you are installing multiple OCT servlet cartridges on the same Internet server. If you are, see Multiple Servlet Scenarios on page 2-19.

15		Physical Path	Enter your local value for the Servlet directory. (See Table 2–4 or Table 2–5.)	Specifies the directory that contains the Java class or archive files used by the JServlet cartridge. This is a full physical pathname.
16		Apply button	Click	Inserts the JServlet cartridge in the OAS application that you just added (*`app_name`*).
17	Success dialog box	OK button	Click	Displays the existing applications in the right frame, now including your new application.

Now you modify the properties of the application and the cartridge to finish setting up your application. See Configuring an Application and JServlet Cartridge on page 2-12.

Configuring an Application and JServlet Cartridge

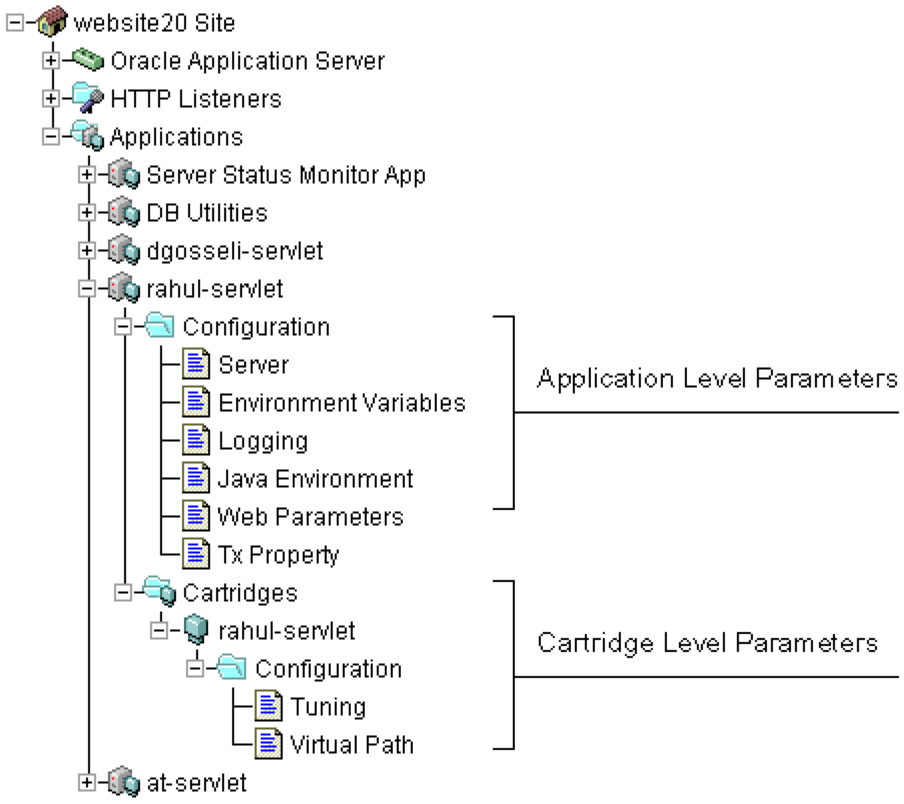
After you have created an application in OAS, and installed a JServlet cartridge in it, you must modify certain configuration parameters for each.

For full details on this task, consult the instructions on application and cartridge configuration in the *Oracle Application Server 4.0 Administration Guide*.

The directory names used here (such as Servlet directory) are explained in Table 2–3 on page 2-3, Table 2–4 on page 2-4, and Table 2–5 on page 2-4.

Figure 2–1 illustrates the location, in the OAS Manager's navigational tree, of the forms for configuring an application and a cartridge. (This figure is provided for convenience; see the OAS *Administration Guide* for details.)

Figure 2–1 OAS Parameter Levels



Configuring Your Application

OAS Application-level parameters apply to all the cartridges in the application.

1. In the OAS navigational tree, open the Applications node.
2. Open your *app_name* application's node.
3. Open the Configuration node under your application's node.

   You should see the following nodes, which are forms for configuring application-level parameters:

   Server

Environment Variables
Logging
Java Environment
Web Parameters
Tx Property

4. Modify these Application-level Configuration parameters on each form, as indicated below. Only the parameters that need to be changed for OCT are described. For other parameters, accept the defaults, or consult other OAS documentation for guidance.

Server Parameters

Accept the defaults, or consult other OAS documentation for guidance.

Environment Variables Parameters

You may need to modify the parameter values as shown in Table 2–11.Be sure to observe the syntax for listing environment variables on your platform:

Windows NT	Directory names written with backslashes (\ ); directory and archive file names separated by semicolons.
Solaris	Directory names written with slashes (/); directory and archive file names separated by colons.

Table 2–11 Environment Variable Parameters

Environment Variable	For Platform	Add To Existing Environment Variable, If Necessary
CLASSPATH	Windows NT	`oct_install/servlet/config.jar`
		`oct_install/servlet/cz3rdpty.jar`
	Solaris	apps.zip
PATH	Windows NT	`oct_install` (or location of Shared Object files)
LD_LIBRARY_PATH	Solaris	`oct_install` (or location of Shared Object files)

Note: For Windows NT, in addition to setting the PATH in OAS, you must add `oct_install` (or the location of the Shared Object files) to the PATH environment variable on the host machine on which the servlet is installed.

Logging Parameters

Modify the parameter values as shown in Table 2–12.

Table 2–12 Application Logging Parameters

Parameter	New Value of Parameter
Logging Directory	*`logging_dir`*
Logging File	*`oas-log`*`.txt` (or your free choice of file name, where *`oas-log`* is different from the filename set for the property `cz.uiservlet.logfilename`.)
Severity Level	15

Java Environment Parameters

Most of these configuration parameters are of the type SYSTEM_PROPERTY. Set them as described under SYSTEM_PROPERTY Java Environment Parameters on page 2-15. Set the other types of configuration parameters according to Table 2–13.

Table 2–13 General Java Environment Parameters

Java Environment Parameter Type	Required Parameter Value
RUNTIME_MODE	`JSERVLET` (Should already be set by default.)
MAX_HEAP	`128M`

There are several Java Environment Parameters (such as ORAWEB_HOME and ORACLE_HOME) that already have default values when you open this form. For these parameters, accept the defaults, or consult other OAS documentation. Do not remove the existing values.

SYSTEM_PROPERTY Java Environment Parameters

The following Java Environment Parameters are specific to OCT. They are all of type SYSTEM_PROPERTY. In the left-hand column of the OAS Java Parameter Configuration form, enter `SYSTEM_PROPERTY` for each parameter; in the right-hand column, enter the appropriate value of the parameter, as illustrated in the syntax and example(s) for each parameter.

In the examples below, the following values are substituted for the placeholders, as shown:

Placeholder	Example Value
hostname	`www.mysite.com`
html_vpath	`/html/`
servlet_vpath	`/servlet/`

The placeholder *servlet_vpath* represents the Virtual Path for the cartridge, which was set in step 14 of Table 2–10, " Summary Steps for Creating an Application and JServlet Cartridge" on page 2-10.

The placeholder *html_vpath* represents the Virtual Path for the HTML directory, which was set in Defining Other Required Virtual Paths on page 2-18.

If there is more than one instance of OAS running on your host machine, then you will have to set a port number with *hostname*. See Multiple OAS Instances on page 2-20.

Tip: When entering a large number of parameters, you can obtain more lines in the OAS form by clicking the Apply button, clicking on another form, then returning to this form.

VERBOSE

Aids in debugging.

Syntax:

```
VERBOSE=[true|false]
```

Example:

```
VERBOSE=true
```

FND_TOP

(Solaris only)

Required for locating the DBC file used for database connectivity. To determine the value to use, enter the command

```
% echo $FND_TOP
```

in a command shell having the desired Oracle Applications environment.

Syntax:

```
FND_TOP=local value of $FND_TOP
```

Example (Solaris only):

```
FND_TOP=/d1/fnd/test
```

Note: For description of the other Java Environment Parameters, you must consult the *Oracle Configurator and SellingPoint Administration Guide*.

Web Parameters

Accept the defaults, or consult other OAS documentation for guidance.

Tx Property Parameters

Accept the defaults, or consult other OAS documentation for guidance.

Configuring Your Cartridge

OAS cartridge-level parameters apply to all the runtime instances of a specific cartridge in an application.

1. In the OAS navigational tree, open the Applications node.
2. Open your `app_name` application's node.
3. Open your application's Cartridges node.
4. Open the `cart_name` node.
5. Open the Configuration node under your cartridge's node.

   You should see the following nodes, which are forms for configuring cartridge-level parameters:

   Tuning
   Virtual Path

6. Modify the cartridge-level Configuration parameters, as indicated below. Only the parameters that need to be changed for OCT are described. For other

parameters, accept the defaults, or consult other OAS documentation for guidance.

Tuning Parameters

Accept the defaults, or consult other OAS documentation for guidance.

Virtual Path Parameters

You may need to modify the parameter values as shown in Table 2–14.

The Virtual Path for the JServlet cartridge itself was set in step 14 of Table 2–10 on page 2-10.

This Virtual Path is referenced elsewhere in these instructions by the placeholder *`servlet_vpath`*.

Table 2–14 Virtual Path Parameters

Virtual Path	Physical Path

`/servlet/`	Examples:
	Windows NT (see Table 2–4):
	`D:\orant\OSP\OSP\`
	Solaris (see Table 2–5):
	`/d1/my-oct/servlet/`

Defining Other Required Virtual Paths

You must also define the virtual paths for the HTML and Media directories. These virtual paths will be referenced in these instructions as *`html_vpath`* and *`media_vpath`*, respectively.

You define the Virtual and File-System Directory values in the Directory form under your www Listener, like this:

1. In the OAS navigational tree, open the HTTP Listeners node.
2. Open the node labelled *your_hostname*:www.
3. Select the Directory node.
4. Enter the values for Virtual Directory and File-System Directory shown in Table 2–15. Click Apply when finished.

Table 2–15 Virtual Directories for the www Listener

Directory	Placeholder	Virtual Directory (normal case)	File-System Directory Example (Solaris)
HTML	*`html_vpath`*	`/html/`	`/d1/my-oct/html/`
Media	*`media_vpath`*	`/media/`	`/d1/my-oct/media/`

Important Note: The indicated values for *`servlet_vpath`*, *`html_vpath`*, and *`media_vpath`* should not be changed unless necessary, because these paths are incorporated into certain of the OCT HTML Template files. Specifying different virtual paths requires you to modify these files. You should not have to set different values unless you are installing multiple OCT servlet cartridges on the same Internet server. If you are, see Multiple JServlet Cartridges on page 2-20.

Reloading the Application Server

You have now finished installing and configuring your Toolkit servlet. In order to use it, you must reload the application server for the new configuration to take effect.

To reload the application server:

1. Select Oracle Application Server in the Oracle Application Server Manager.

   This displays a list of Oracle Application Server Processes in the right frame.
2. Select “ALL”.
3. Click ⏮ to reload the application server.

Multiple Servlet Scenarios

The instructions given elsewhere in this chapter assume that you are installing a single cartridge for the OCT UI Servlet in a single instance of OAS. This section describes some possible alternatives that may be necessary for your situation:

- Multiple OAS Instances on page 2-20

- Multiple JServlet Cartridges on page 2-20

Multiple OAS Instances

It is possible that you may want to install more than one instance of OAS on the same host machine. If you do, then you need to change some basic configuration settings.

For background, see the *Oracle Application Server 4.0 Administration Guide*, especially the chapter on managing and configuring HTTP listeners.

The default port number for the OAS Node Manager listener is 8888; the default port number for the OAS www listener is 80. If you have installed only one OAS instance on your Internet server host machine, then you do *not* need to specify these default port numbers when specifying a URL with OAS, such as this system property:

```
cz.uiservlet.templateurl=http://www.mysite.com/html/US/czFraNS.htm
```

However, if you install another OAS instance on your Internet server host machine, then you must set different port numbers for the OAS Node Manager and www listeners. You must then use these port numbers in URLs in OCT parameters for cartridges installed in that instance. For example, if an OAS Node Manager listener uses port 7777, and the www listener uses port 70, then all the URLs must reflect this, as shown:

```
cz.uiservlet.templateurl=http://www.mysite.com:70/html/US/czFraNS.htm
```

Multiple JServlet Cartridges

It is possible that you may want to install multiple versions of the OCT UI Servlet cartridge on your Internet server. For instance, you may wish to maintain “development” and “production” versions.

OAS does not allow you to create duplicate Virtual Path names for different cartridges anywhere on a single host machine. You must provide unique Virtual Path names for each cartridge.

This might not appear to be an impediment, but certain of the OCT HTML Template files incorporate the cartridge’s Virtual Path in order to locate the UI Servlet.

Warning: If you change a new cartridge's Virtual Path in order to be unique, without making other adjustments, the OCT HTML Template files will not work.

In order to deal with this situation, you should use this method:

- Set up alternate virtual and physical paths for the OCT Servlet, HTML, and Media directories
- Edit the HTML Template files to reflect this structure

These tasks are described here.

Setting Up Alternate Paths

Set up a physical directory structure that parallels the one created by the OCT installation (which is shown in Table 2–3, " General Structure of Directories for OCT" on page 2-3, with platform-specific examples in Table 2–4 and Table 2–5).

Assuming that you wish to set up "Development" and "Production" versions, Table 2–16 shows a suggested structure (for Solaris):

Table 2–16 Parallel Directory Structures

Directory	"Development"	"Production"
Servlet	`/d1/my-oct/dev-servlet/`	`/d1/my-oct/prod-servlet/`
HTML	`/d1/my-oct/dev-html/`	`/d1/my-oct/prod-html/`
Media	`/d1/my-oct/dev-media/`	`/d1/my-oct/prod-media/`

Now define the OAS virtual paths to match these locations, as shown in Table 2–17:

Table 2–17 Parallel Virtual Paths

Defined for	Virtual Path/ Virtual Directory	File-System Directory	Placeholder
	"Development"		
Cartridge	`/dev-servlet/`	`/d1/my-oct/dev-servlet/`	*`servlet_vpath`*
www Listener	`/dev-html/`	`/d1/my-oct/dev-html/`	*`html_vpath`*
www Listener	`/dev-media/`	`/d1/my-oct/dev-media/`	*`media_vpath`*

Table 2–17 (Cont.) Parallel Virtual Paths

Defined for	Virtual Path/ Virtual Directory	File-System Directory	Placeholder
	"Production"		
Cartridge	`/prod-servlet/`	`/d1/my-oct/prod-servlet/`	*`servlet_vpath`*
www Listener	`/prod-html/`	`/d1/my-oct/prod-html/`	*`html_vpath`*
www Listener	`/prod-media/`	`/d1/my-oct/prod-media/`	*`media_vpath`*

The placeholders (such as *`servlet_vpath`*) are included here because they are referenced elsewhere in this document.

You define the Virtual Path for a cartridge as described in Virtual Path Parameters on page 2-18.

You define the Virtual and File-System Directory values in the Directory form under your www Listener, like this:

1. In the OAS navigational tree, open the HTTP Listeners node.
2. Open the node labelled *your_hostname*:www.
3. Select the Directory node.
4. Enter the values for Virtual Directory and File-System Directory shown in Table 2–17. Click Apply when finished.

Editing the HTML Template Files

After you have set up your parallel virtual paths and directories, edit the files listed in Table 2–18, according to Table 2–19. These files are installed in your HTML directory.

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

Table 2–18 HTML Template Files To Be Edited

File	Example of text to be changed
czCntnt.htm	`action="/`**`servlet`**`/oracle.apps.cz.servlet.UiServlet"`

Table 2–18 (Cont.) HTML Template Files To Be Edited

File	Example of text to be changed
czFraIE.htm	`src="/`**`servlet`**`/oracle.apps.cz.servlet.Proxy"`
czFraNS.htm	`src="/`**`servlet`**`/oracle.apps.cz.servlet.Proxy"`
czHeader.htm	`src="/`**`media`**`/czSumm0.gif"`
czSource.htm	`var IMAGESPATH = '/`**`media`**`/';`
czUIBkr.js *	`doc.location = '/`**`html`**`/czCntnt.htm';`

Table 2–19 HTML Template File Substitutions

Replace this text	With this text	Placeholder
	“Development”	
`/servlet/`	`/dev-servlet/`	*`servlet_vpath`*
`/html/`	`/dev-html/`	*`html_vpath`*
`/media/`	`/dev-media/`	*`media_vpath`*
	“Production”	
`/servlet/`	`/prod-servlet/`	*`servlet_vpath`*
`/html/`	`/prod-html/`	*`html_vpath`*
`/media/`	`/prod-media/`	*`media_vpath`*

The placeholders (such as *`servlet_vpath`*) are included here because they are referenced elsewhere in this document.

***** To modify `czUIBkr.js`, edit `czAll.js`, and search for the string “czUIBkr.js”, to locate the code that applies to `czUIBkr.js`. After modifying the location for `/html/`, and saving your customized version of `czAll.js`, use `czAll.js` to make a new version of `czJSSrc.jar`.

3 Integration: Session Initialization

Overview of this Chapter

How it Works

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

- Import its data into the Oracle Configurator schema so that it is available to Oracle Configurator Developer. Importing is described in the *Oracle Configurator and SellingPoint Administration 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.

Toolkit provides you with:

- A **servlet**, the UI Servlet, that handles interaction between the host application, the configuration window, and the product Model that you define in Configurator Developer (which contains product structure, logic rules, and user interface definitions).
- The host application invokes the URL of the UI 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 are described in this chapter.
- A set of **HTML Template files** that interact with the UI 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 Toolkit Architecture on page 1-2 in Chapter 1, "Introduction" for an explanation of the interaction between all these elements.

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 UI Servlet.

You must first install the OCT UI Servlet, as described in Chapter 2, "Installing the Servlet".

There are two basic situations that the host application must handle when integrating the configuration window:

You need to handle this...	As described in...
Initialization of the configuration window, to prepare it for your user to perform configuration selections.	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.	Definition of Session Termination on page 4-1

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.

Definition of Session Initialization

Session initialization takes place when your host application invokes the UI 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 Initialization Parameter Types on page 3-7.

When your host application invokes the configuration window, the initialization message is sent to the UI 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.

Setting Parameters

You specify `<initialize>` and its parameters as the value of an XML message that is passed to the UI Servlet. The UI Servlet is invoked through its URL, which is determined when you install it (as described in Creating an Application and JServlet Cartridge on page 2-9). By default, the URL is `/servlet/oracle.apps.cz.servlet.UiServlet`.

Basic 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 UI 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 emphasized.

Example 3–1 Syntax of intialization message in HTML context

```
...
<form action="/servlet/oracle.apps.cz.servlet.UiServlet" 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 Now">
</form>
...
```

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

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>"
```

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 3-7.

See Initialization Parameter Descriptions on page 3-14 for the complete list of valid parameters to the initialization message.

Example 3–2 Basic XML initialization parameters

```
<initialize>
    <param name="two_task">vis11</param>
    <param name="gwyuid">applsyspub</param>
    <param name="user">mfg</param>
    <param name="pwd">welcome</param>
    <param name="ui_type">DHTML</param>
    <param name="ui_def_id">1380</param>
    <param name="return_url">http://server01/servlet/Checkout</param>
</initialize>
```

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

Parameter type	Name	Value	Description
Login	`two_task`	vis11	The name of the database instance to connect to. This value can be read from the environment variable TWO_TASK.
Login	gwyuid	applsyspub	The gateway user ID required for authentication with the Oracle Applications protocol.
Login	user	mfg	The user ID of the login user.
Login	pwd	welcome	The password of the login user.
Login	ui_type	DHTML	The type of web pages to be returned by the UI Servlet. The configuration window requires that this be set to “DHTML”.

Table 3–1 (Cont.) Explanation of initialization parameters in Example 3–2

Parameter type	Name	Value	Description
Configuration	ui_def_id	1380	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 Configuration Identification Parameters on page 3-8.
Return	`return_url`	(a URL)	The URL of the Java servlet that processes the configurator's termination message.

Minimal Test of Initialization

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

This example includes the invocation of the UI Servlet shown in Example 3–1 and the initialization message parameters 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="/servlet/oracle.apps.cz.servlet.UiServlet" method="post" >
<input type=hidden name="XMLmsg" value=
"<initialize>
    <param name="two_task">vis11</param>
    <param name="gwyuid">applsyspub</param>
    <param name="user">mfg</param>
    <param name="pwd">welcome</param>
    <param name="ui_type">DHTML</param>
    <param name="ui_def_id">1380</param>
    <param name="return_url">http://server01/servlet/Checkout</param>
</initialize>">
<input type="submit" value="Configure Now">
</form>
</body>
</html>
```

Parameter Validation

When your host application invokes the UI 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 Configuration Identification Parameters on page 3-8.

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

Initialization Parameter Types

This section describes the use of the types of initialization parameters listed in Table 3–2 on page 3-7. All of the initialization parameters are described alphabetically in Initialization Parameter Descriptions on page 3-14.

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.	Connection Parameters on page 3-8
Configuration	Yes	Identification of the model to be configured, or of the existing configuration to be modified. See Configuration Identification Parameters on page 3-8.	Configuration Identification Parameters on page 3-8
Return	Yes	Identification of the return URL that handles the results from the configuration window, such as configuration outputs.	Return URL Parameter on page 3-10
Pricing and ATP	No	Identification of the procedures and interfaces to be used for obtaining prices and ATP dates.	Pricing Parameters on page 3-10 ATP Parameters on page 3-12

Table 3–2 (Cont.) Types of Initialization Parameters

Type	Required?	Description	See
Other	No	Miscellaneous desired information.	Arbitrary Parameters on page 3-14

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_ticket
- alt_database_name, user, and pwd
- two_task, user, pwd, gwyuid, and fndnam

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

Configuration Identification Parameters

There are several different ways in which you can identify the model to be configured, or the existing configuration to be modified. You must use one of the following parameters or combinations of parameters in your initialization message:

Table 3–3 Configuration Identification Parameters

Method for Configuration Identification	Initialization Parameters
Identifying the User Interface Definition	ui_def_id
Identifying the Configuration	config_header_id
	config_rev_nbr
Identifying the Model	context_org_id
	model_id
	config_creation_date

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

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).

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 Oracle SellingPoint application. The configuration revision number distinguishes among particular saved configurations sharing the same header information.

Identifying the Model

Parameters to specify:

- context_org_id
- model_id
- config_creation_date

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.

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 closed the configuration window.

- return_url

```
<param name="return_url">http://server01/servlet/Checkout</param>
```

The example parameter above comes from Example 3–3, "Minimal HTML for invoking the configuration window" on page 3-6.

The URL specification in the `return_url` parameter must end with the name of the servlet class. You cannot successfully pass parameters to the class in this URL. 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 return servlet is installed in your web server's Servlet directory, as described in Files for the Servlet Directory on page 2-5 in Chapter 2, "Installing the Servlet".

See The Return URL on page 4-10 for details on the implementation of the return servlet.

Pricing Parameters

See Chapter 5, "Integration: Pricing and ATP" for details on the use of these parameters. See Appendix C, "Examples of Pricing and ATP Callback Procedures" on page C-1 for examples.

Choose from two distinct sets of pricing parameters, depending on which pricing methods are required for your host application:

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 Oracle Applications Release 11i Pricing Parameters on page 3-11

Pricing method	Use these parameters...
Oracle Applications pricing (using a price list ID)	price_list_id and the associated parameters listed under Oracle Applications Release 10.7 and 11.0 Pricing Parameters on page 3-12

If the Oracle Configurator is running in one database (e.g., Release 11*i*), and connecting to another database (e.g., Release 10.7/11.0) to perform pricing, you need to provide the apps_connection_info parameter.

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

Pricing Parameter Precedence

Your initialization message can contain a mix of both types of parameters, but only one set will be used.

- If pricing_package_name is provided in your initialization message, then price_list_id is ignored, even if provided.
- price_mult_items_proc takes precedence over price_single_item_proc, if both are provided.
- price_list_id is required in order to use Release 10.7 and 11.0 pricing.
- If neither pricing_package_name nor price_list_id is provided, an error is raised.

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 or price_single_item_proc

See Pricing Parameter Precedence on page 3-11.

Oracle Applications Release 10.7 and 11.0 Pricing Parameters

To use Oracle Applications pricing, provide these parameters:

- price_list_id
- and one or more of the following parameters:
  - agreement_id
  - agreement_type_code
  - customer_id
  - gsa
  - invoice_to_site_use_id
  - order_type_id
  - po_number
  - pricing_flex_field parameters
  - ship_to_site_use_id

See Pricing Parameter Precedence on page 3-11.

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*.

ATP Parameters

See Chapter 5, "Integration: Pricing and ATP" for details on the use of these parameters. See Appendix C, "Examples of Pricing and ATP Callback Procedures" on page C-1 for examples.

Choose from two distinct sets of ATP (Available To Promise) parameters, depending on which ATP methods are required for your host application:

If the Oracle Configurator is running in one database (e.g., Release 11*i*), and connecting to another database (e.g., Release 10.7/11.0) to perform ATP calculations, you need to provide the apps_connection_info parameter.

For descriptions of the individual parameters, see Initialization Parameter Descriptions on page 3-14.

ATP Method Precedence

Your initialization message can contain a mix of both types of parameters, but only one set will be used.

If the callback parameters are provided in your initialization message, then they take precedence, and the 10.7/11.0 parameters are ignored, even if provided.

If neither the callback parameters nor the 10.7/11.0 parameters are provided, an error is raised.

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

Oracle Applications Release 10.7 and 11.0 ATP Parameters

To use existing Oracle Applications ATP procedures, provide all of these parameters:

- application_id
- atp_timeout
- responsibility_id
- user_id

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 Basic 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 more information.)

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

Initialization Parameter Descriptions

This section lists alphabetically all the parameters of the initialization message, which is described in Setting Parameters on page 3-4.

agreement_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

agreement_type_code
For use with Oracle Applications 10.7 and 11.0 Pricing.

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.

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 (e.g., Release 10.7/11.0) 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_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 UI 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`.

atp_timeout
The maximum number of seconds that the Oracle Applications 10.7/11.0 ATP calculation method should wait for a result from the server.

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

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. This date must be in the format "MM-DD- YYYY" where YYYY is a four digit year, MM is a two digit month and DD is a two digit day of the month.

config_header_id
The identifier for an existing configuration. Only used for retrieving a configuration previously saved by the Oracle SellingPoint application. 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_rev_nbr
The configuration revision number. Only used for retrieving a configuration previously saved by the Oracle SellingPoint application. 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.

customer_id
For use with Oracle Applications 10.7 and 11.0 Pricing. The customer ID from the header of the host application.

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.

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.

gsa
For use with Oracle Applications 10.7 and 11.0 Pricing.

gwyuid
Used to establish an Oracle Applications context. Its value can be read from the Forms environment variable GWYUID.

icx_ticket
An icx_ticket encodes an Oracle Applications session. From a Forms session, the host application can call `icx_sec.createSession` and then `icx_sec.encrypt3` on the `session_id` that is returned from `createSession`. This is the recommended way for Oracle Applications applications to call the configuration window. When passing an icx_ticket, the host application must also pass a `database_id`.

invoice_to_site_use_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

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.

model_quantity
The value of this parameter is a number that indicates how many of the model are being configured. If not specified, it defaults to 1. 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.

order_type_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

po_number
For use with Oracle Applications 10.7 and 11.0 Pricing.

price_list_id
The identifier for the price list. If not present, pricing information using the Oracle Applications 10.7/11.0 calls will not be available.

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; either this parameter or `price_single_item_proc` must be provided if pricing callbacks are to be used. This procedure 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; either this parameter or `price_mult_items_proc` must be provided if pricing callbacks are to be used. This procedure will not be called if `price_mult_items_proc` is provided.

pricing_*flex_field*
For use with Oracle Applications 10.7 and 11.0 Pricing. The pricing flexfield values for the model to be configured. These parameters are named `pricing_attribute1` through `pricing_attribute15`.

pricing_package_name
The name of the PL/SQL interface package that the UI 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`.

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.

read_only
If the value is "true", the UI Server provides a read-only UI for viewing configurations.

requested_date
When getting ATP dates, the requested date entered on the order line. The format of the date must be "MM-dd-yyyy.

responsibility_id
The ID from FND_RESPONSIBILITY.RESPONSIBILITY_ID.

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. See Return URL Parameter on page 3-10 for details.

save_config_behavior
Value is one of:

`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.
`overwrite`	The existing configuration header and revision will be used.

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

save_usage_behavior
Value is one of:

`never`	A usage will not be saved.
`new_usage`	A new usage will be saved if there is none, and an existing usage will be updated if found.

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.

ship_to_site_use_id
For use with Oracle Applications 10.7 and 11.0 Pricing.

template_url
A URL to 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 `html/US`, when specifying this parameter.

terminate_msg_behavior

Value is one of:

`full`	The entire terminate message is passed back to the host application.
`brief`	No output or messages are passed to the caller.

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. Must be set to `DHTML` when using the Toolkit configuration window.

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.

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 Integration: Session Termination

How it Works

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

What You Do

See What You Do on page 3-2, in Chapter 3.

Definition of Session Termination

Session termination takes place when the configuration window is closed by one of these conditions:

Table 4–1 Termination conditions

Condition	Example	Explanation
Submission	Your user clicks the Done button.	See Submission on page 4-3
Cancellation	Your user clicks the Cancel button.	See Cancellation on page 4-8
Error	A connection cannot be made to the database.	See Error on page 4-8
Timeout	The configuration window has been inactive for a specified length of time.	See Session Timeout on page 4-9

When the configuration window is closed, terminating your user's configuration session, the UI 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 XML Message Structure on page 4-2.

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>
```

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 Customizing the Header Frame on page 6-6, in Chapter 6, "User Interface".

When the Done button is clicked, the UI 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 The Return URL on page 4-10

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:

- Configuration Status
- Configuration Outputs

- Configuration Messages

Configuration Status

The current configuration status is described by the following subelements of `<terminate>`:

<config_header_id>
The main identifier of an existing configuration. See the description for config_header_id on page 3-15. 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 3-16. 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>
Value is one of:

`save`	If the configuration was saved.
`cancel`	If the configuration was cancelled.
`error`	If an error was detected while executing in the UI Server.
`timeout`	If the UI Server timed out the session because of inactivity.
`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. If the value of this element is:

`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.

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. Here is an example of configuration outputs in the termination message:

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>
```

<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 ATP Parameters on page 3-12, and Chapter 5, "Integration: Pricing and ATP" on page 5-1.

<list_price>
Contains the list price for the selected option. This is calculated by using the pricing procedure specified in the initialization message. See Pricing Parameters on page 3-10, and Chapter 5, "Integration: Pricing and ATP" on page 5-1.

<discounted_price>
Contains the discounted price for the selected option. This is calculated by using the pricing procedure specified in the initialization message. See Pricing Parameters on page 3-10, and Chapter 5, "Integration: Pricing and ATP" on page 5-1.

Configuration Messages

The messages generated by the UI 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 UI 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>
```

<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.

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>
```

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 Connection Parameters on page 3-8)
- Lack of any configuration parameters (see Configuration Identification Parameters on page 3-8)
- 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 UI 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>
```

Session Timeout

Session timeout occurs when the configuration window has been inactive for a specified length of time (for instance, if the user has not made any selections in it).

In Toolkit, sessions in the configuration window are constantly checked for timeout, by a scavenger thread.

You control the behavior of this thread by setting certain system properties. In Oracle Application Server, you do this by configuring Java Environment Parameters of the type SYSTEM_PROPERTY. See Java Environment Parameters on page 2-15.

System Property	
cz.uiserver.wakeintervalseconds	The number of seconds between executions of the UI Server's session timeout thread.
cz.uiserver.timeoutseconds	The number of seconds that a session is inactive before it is timed out by the UI Server's session timeout thread.

The following settings would cause the thread to run every 8 minutes, checking for sessions that have been inactive for 20 minutes or longer:

```
cz.uiserver.wakeintervalseconds=480

cz.uiserver.timeoutseconds=1200
```

When a session times out the UiServer creates a termination message with the `timeout` keyword for the <exit> element:

```
<terminate>
  <exit>timeout</exit>
</terminate>
```

After the UI Server generates the timeout message, it notifies the UI Servlet, which sends this termination message back to the host application through the `return_url`, if the host application specified one in the initialization message. If the UI

Servlet then attempts to process another event for a configuration window session that has already timed out, the following XML message is generated:

```
<terminate>
  <exit>error</exit>
  <config_messages>
    <message>
      <message_type>error</message_type>
      <message_text>SESSION-NAME- not found and presumed to have timed
out.</message_text>
    </message>
  </config_messages>
</terminate>
```

This message will also be generate if the session name is incorrect.

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.

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://server01/servlet/Checkout</param>
```

The example parameter above comes from Example 3–3, "Minimal HTML for invoking the configuration window" on page 3-6.

See also:

- Return URL Parameter on page 3-10
- return_url on page 3-19
- Basic Parameter Syntax on page 3-4

Implementing the Return URL

See Example B–1 in Appendix B 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 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 Integration: Pricing and ATP

How it Works

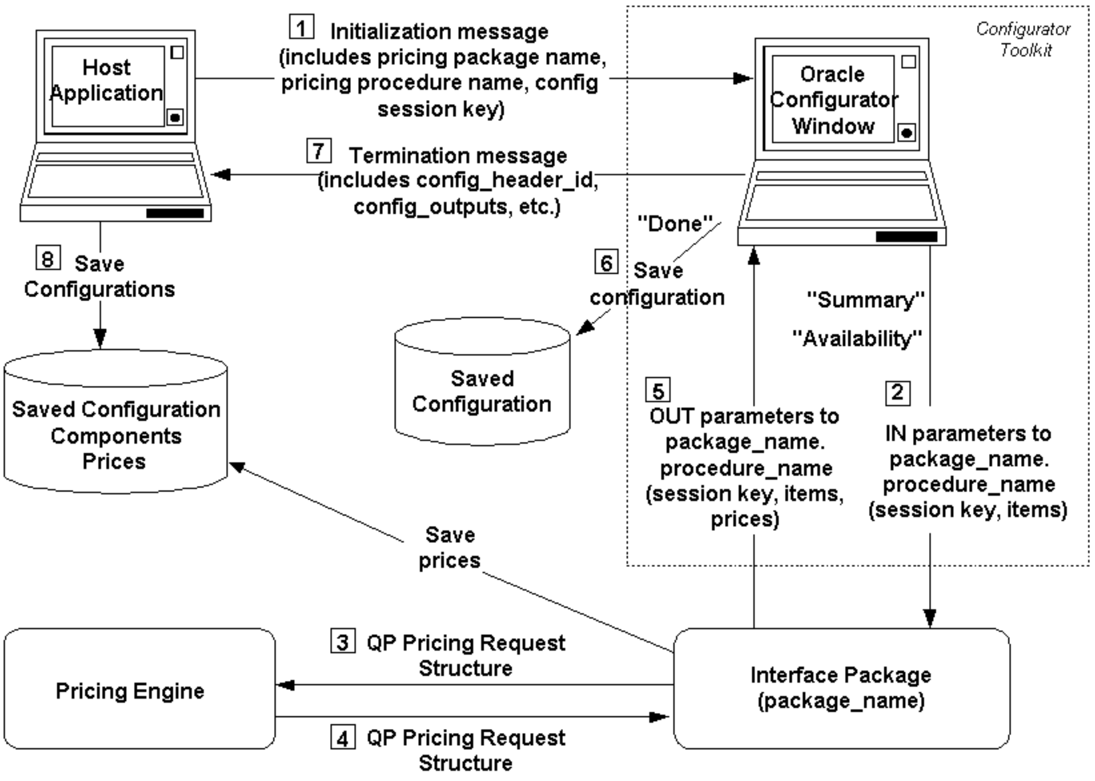
Toolkit allows your host application to display pricing and ATP (Available To Promise) information in the OCT configuration window. There is more than one method for presenting this information, to account for these factors:

- The OCT configuration window can be called from a variety of different host applications, both Oracle (such as iStore) and non-Oracle.
- Pricing and ATP data can be obtained from either an Oracle Applications 10.7/11.0 database, or an Oracle Applications Release 11*i* database.
- 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 Toolkit, and Toolkit does not call the host application's pricing and ATP procedures directly. Instead, Toolkit 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 OCT configuration window at initialization.

Figure 5–1 shows the architecture for the pricing mechanism in Toolkit. (The architecture for ATP is analogous.)

Figure 5–1 Pricing Architecture in Toolkit


Host Application
1 Initialization message (includes pricing package name, pricing procedure name, config session key)
Configurator Toolkit
Oracle Configurator Window
7 Termination message (includes config_header_id, config_outputs, etc.)
"Done"
8 Save Configurations
6 Save configuration
"Summary"
"Availability"
Saved Configuration
Saved Configuration Components Prices
5 OUT parameters to package_name. procedure_name (session key, items, prices)
2 IN parameters to package_name. procedure_name (session key, items)
Save prices
Pricing Engine
3 QP Pricing Request Structure
4 QP Pricing Request Structure
Interface Package (package_name)

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

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 UI Servlet, passing as initialization parameters the names of your packages and procedures.

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

1. Install the OCT interface packages in your database, by running the appropriate scripts. See Database Compatibility on page 5-3 and Installation Scripts on page 5-5.
2. Write your own PL/SQL pricing or ATP procedures, using the OCT interfaces. See the *Oracle Configurator and SellingPoint Administration Guide*, and Implementation Scripts on page 5-4 for details. See Examples of Pricing and ATP Callback Procedures on page C-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 OCT UI Servlet, provide parameters that specify the name of the pricing package, and the procedure to use. See Initialization Parameters on page 5-8 for an example, and see Pricing Parameters on page 3-10 and ATP Parameters on page 3-12 for explanation of the parameters.

Database Compatibility

Oracle Configurator Toolkit 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 “direct import” script. See Installation Scripts on page 5-5 and the *Oracle Configurator and SellingPoint Administration Guide*.

There are several likely scenarios for pricing and ATP integration:

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 pricing to Order Management (Oracle Applications Release 11*i*), you write custom programs in your host application.

To Integrate with...	You would...
Oracle Applications 10.7/11.0 database	You cannot use the QP Advanced Pricing engine. You must use a BOM model to perform pricing. To import BOM data to the Oracle Configurator schema tables, you run "direct import" scripts. To export pricing data to Order Entry (Oracle Applications Release 10.7/11.0), you write custom programs, using the SO_INTERFACE_* tables. In OCT, you pass the price_list_id parameter in your initialization message to invoke pricing.
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.

Implementation Scripts

These are the scripts that must be run to make the Oracle Configurator pricing and ATP interfaces available. You do not normally run these scripts directly. They are run by the scripts listed in Installation Scripts on page 5-5. They are listed here for your information. Consult the *Oracle Configurator and SellingPoint Administration Guide* for details on when and how to run scripts.

- CZ_LIST_PRICE_B.sql
- CZ_LIST_PRICE_S.sql
- CZ_PRC_UTIL_B.sql
- CZ_PRC_UTIL_S.sql
- CZ_PRC_CALLBACK_UTIL_B.sql
- CZ_PRC_CALLBACK_UTIL_S.sql

- CZ_ATP_UTIL_B.sql
- CZ_ATP_UTIL_S.sql
- CZ_ATP_CALLBACK_UTIL_B.sql
- CZ_ATP_CALLBACK_UTIL_S.sql
- CZ_ATP_CALLBACK_UTIL_B_80.sql

- CZ_PRC_CALLBACK_UTIL_B_80.sql

Installation Scripts

These are the scripts that *install* the implementation of the Oracle Configurator pricing and ATP interfaces. Consult the *Oracle Configurator and SellingPoint Administration Guide* for details on when and how to run them. These scripts run certain of the scripts listed in Implementation Scripts on page 5-4.

- DBAdmin\GO_IMPORT.sql
- DBAdmin\InstAppsIntegrate.sql
- DBAdmin\InstAppsIntegrateViaLink.sql
- DBAdmin\Server\13i_to_14a_Server.sql
- DBAdmin\Server\4.2_to_4.2.1_Server.sql
- DBAdmin\Server\14a_to_14b_server.sql
- DBAdmin\Server\CZ_SERVER.sql
- DBAdmin\Server\UPGRADE_SERVER.sql

The Pricing Callback Interface

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

- Price Single Item
- Price Multiple Items

The Price Single Item procedure returns price information for a single item. Table 5–1 describes the parameters for the Price Single Item procedure.

Table 5–1 Price Single Item Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Varchar2(50)	Required	
price_type	In		Required	Values are: 'LIST', 'SELLING', 'BOTH'
ps_node_id	In	Number	Conditionally Required	PS_Node_ID or BOM_Item_Key is required

Table 5–1 (Cont.) Price Single Item Procedure Parameters

Parameter	In/Out	Type	Required	Note
item_key	In	Varchar2	Conditionally Required	PS_Node_ID or Item_Key is required. If models are imported from Oracle BOM, this key is COMPONENT_CODE:EXPLOSION_TYPE:ORGANIZATION_ID:TOP_ITEM_ID
quantity	In	Number	Required	
uom_code	In	Number	Required	
list_price	Out	Number	n/a	
selling_price	Out	Number	n/a	
msg_data	Out	Varchar2	n/a	

The Price Multiple Items procedure returns price information for a group of items. Table 5–2 describes the parameters for the Price Multiple Items procedure.

Table 5–2 Price Multiple Items Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Number	Required	
price_type	In	Varchar2	Required	Values are: 'LIST', 'SELLING', 'BOTH'
config_total_price	Out	Number	n/a	

Example 5–1 on page 5-7 shows a possible implementation of the callback interface for both single and multiple-item pricing procedures.

Example C–1 on page C-1 shows a minimal example of the use of the callback interface.

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 5–1 and Table 5–2.

Example 5–1 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;
```

The ATP Callback Interface

The “Get ATP Dates” procedure returns availability dates for all selected options and for the configuration as a whole. Table 5–1 describes the parameters for the Get ATP Dates procedure.

Table 5–3 ATP Procedure Parameters

Parameter	In/Out	Type	Required	Note
configurator_session_key	In	Varchar2(50)	Required	
warehouse_id	In		Conditionally Required	warehouse_id, ship_to_org_id, or customer_id and customer_site_id is required
ship_to_org_id	In	Number	Conditionally Required	
customer_id	In	Number	Conditionally Required	
customer_site_id	In	Number	Conditionally Required	
requested_date	In	Date		

Table 5–3 (Cont.) ATP Procedure Parameters

Parameter	In/Out	Type	Required	Note
ship_to_group_date	Out	Date		

Example 5–2 on page 5-8 shows an implementation of the callback interface for ATP procedures.

See Example C–3, "Example of Callback ATP Procedure" on page C-2 for an example in context.

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 5–3.

Example 5–2 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;
```

Initialization Parameters

Example 5–3 on page 5-8 shows how you would specify pricing and ATP parameters in your initialization message. The names and values of the pricing and ATP parameters are typographically emphasized.

Example 5–3 Initialization Message Using Pricing and ATP Parameters

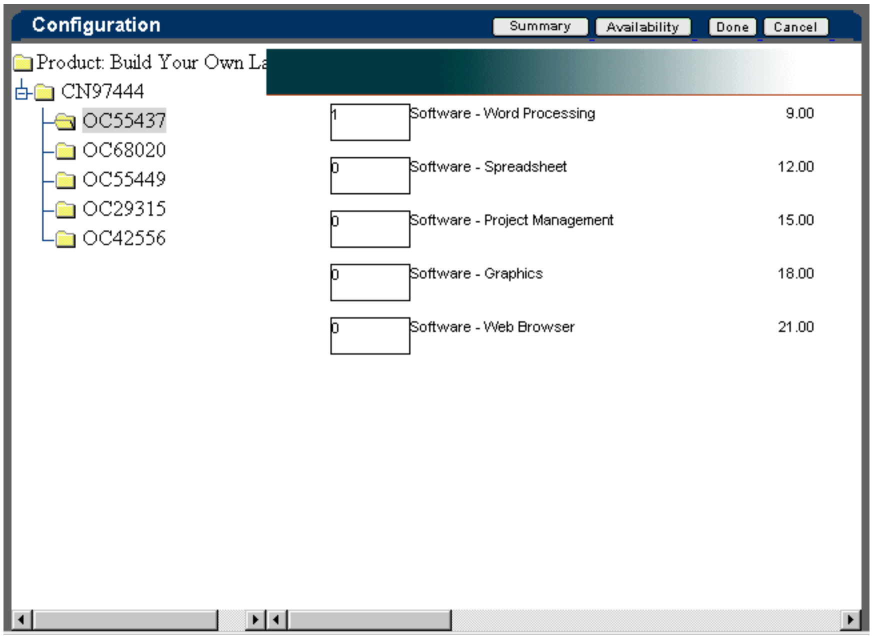
```
<html>
<head>
<title>Project Test</title>
</head>
<form action="http://www.mysite.com:60/servlet/oracle.apps.cz.servlet.UiServlet"
method="post" ><input type="hidden" name="XMLmsg" value='<initialize>
<param name="alt_database_name">vis11</param>
```

```
<param name="user">mfg</param>
<param name="pwd">welcome</param>
<param name="ui_type">DHTML</param>
<param name="ui_def_id">1632</param>
<param name="price_list_id">1040</param>
<param name="pricing_package_name">cz_price_test</param>
<param name="price_mult_items_proc">price_multiple_items</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="application_id">300</param>
<param name="responsibility_id">20559</param>
<param name="user_id">1068</param>
<param name="atp_timeout">10</param>
<param name="customer_id">1000</param>
</initialize>'></form>
<br>
Loading configurator...
</body>
</html>
```

Testing Pricing and ATP Integration

Figure 5–2 on page 5-10 shows the effect of the initialization message shown in Example 5–3 on page 5-8.

Figure 5–2 Pricing Data in the Configuration Window

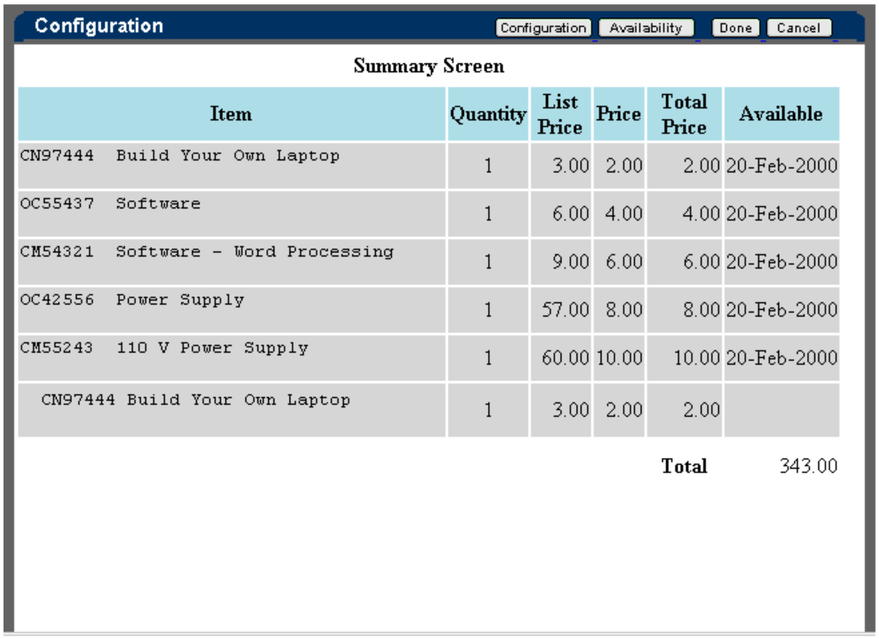


1	Software - Word Processing	9.00
0	Software - Spreadsheet	12.00
0	Software - Project Management	15.00
0	Software - Graphics	18.00
0	Software - Web Browser	21.00

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

Figure 5–3 on page 5-11 shows how pricing and ATP data is displayed when you click the Summary button.

Figure 5–3 Pricing and ATP Data in the Summary Pane



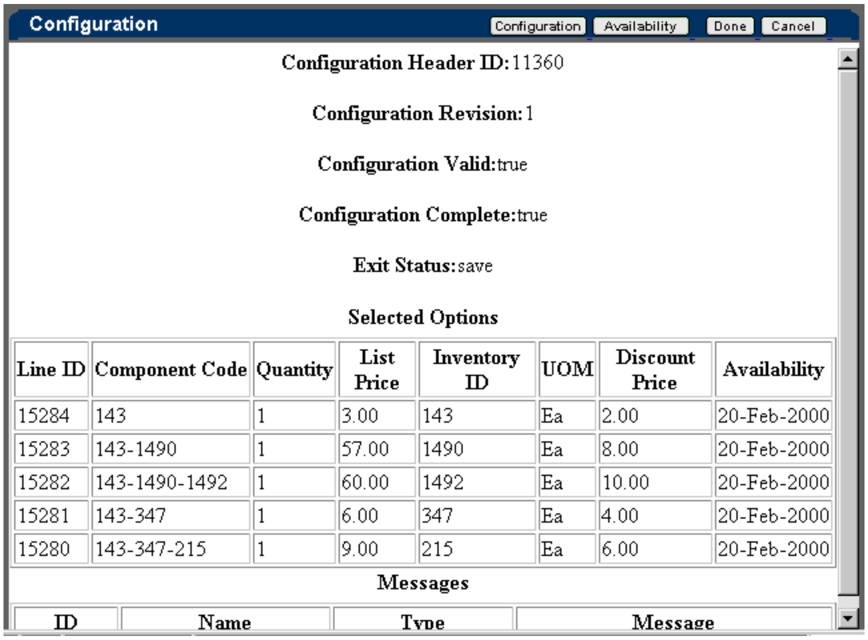
Summary Screen

Item	Quantity	List Price	Price	Total Price	Available
CN97444 Build Your Own Laptop	1	3.00	2.00	2.00	20-Feb-2000
OC55437 Software	1	6.00	4.00	4.00	20-Feb-2000
CM54321 Software - Word Processing	1	9.00	6.00	6.00	20-Feb-2000
OC42556 Power Supply	1	57.00	8.00	8.00	20-Feb-2000
CM55243 110 V Power Supply	1	60.00	10.00	10.00	20-Feb-2000
CN97444 Build Your Own Laptop	1	3.00	2.00	2.00	
				Total	343.00

Your pricing and ATP procedures are used to calculate the selling price 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.

Figure 5–4 on page 5-12 shows how pricing and ATP data is displayed when you click the Done button. See Submission on page 4-3 for an explanation of the data provided.

Figure 5–4 Pricing and ATP Data in the Submission Display



Configuration Header ID: 11360

Configuration Revision: 1

Configuration Valid: true

Configuration Complete: true

Exit Status: save

Selected Options

Line ID	Component Code	Quantity	List Price	Inventory ID	UOM	Discount Price	Availability
15284	143	1	3.00	143	Ea	2.00	20-Feb-2000
15283	143-1490	1	57.00	1490	Ea	8.00	20-Feb-2000
15282	143-1490-1492	1	60.00	1492	Ea	10.00	20-Feb-2000
15281	143-347	1	6.00	347	Ea	4.00	20-Feb-2000
15280	143-347-215	1	9.00	215	Ea	6.00	20-Feb-2000

Messages

ID	Name	Type	Message

6 User Interface

Oracle Configurator Toolkit 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.

Toolkit 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.

How it Works

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

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.

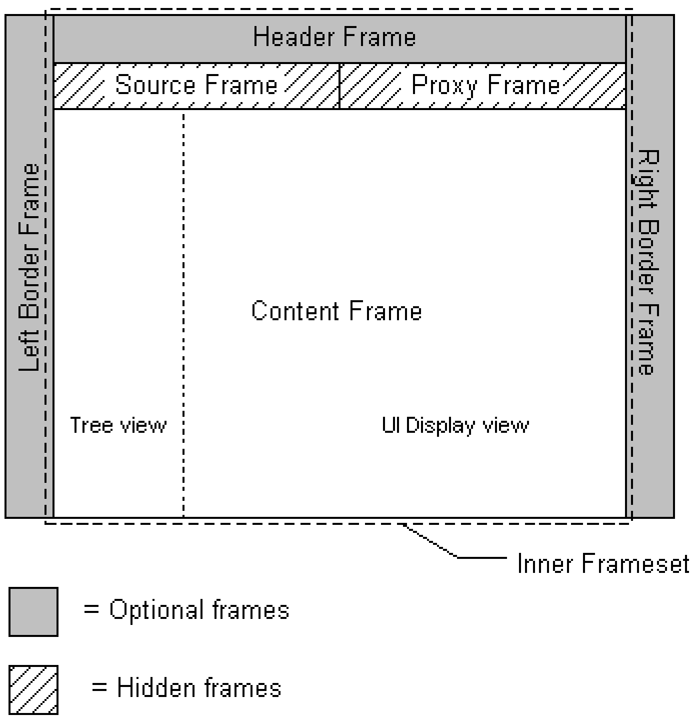
The Template is divided into two areas:

- 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. The structure of the template, as shown in Figure 6–1 on page 6-2, is:

- the optional, customizable Outer Frameset, which consists of:
  - the required Inner Frameset
  - the optional, customizable Left and Right Border Frames
- the required, non-customizable Inner Frameset, which consists of:
  - the optional Header Frame
  - the required Source and Proxy Frames
  - the required Content Frame

Figure 6–1 The Structure of the HTML Template



The Source and Proxy 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 6–1.

Table 6–1 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 and SellingPoint Administration Guide*. You can also specify the frameset file in the initialization message for the UI Servlet, using the parameter template_url on page 3-19.

The constraints on customizing the Template are summarized in Table 6–2.

Table 6–2 Constraints on the HTML Template

Required?	Customize?	Frame/Frameset			
No	Yes	Outer Frameset			
No	Yes		Left Border Frame		
Yes	No		Inner Frameset		
No	Yes			Header Frame	
Yes	No			Source and Proxy Frameset	
Yes	No				Source Frame
Yes	No				Proxy Frame
Yes	No			Content Frame	
No	Yes		Right Border Frame		

The Outer Frameset divides the hosting frame into three columns so that you can have a left and right border.

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.

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

For an example of how the HTML Template appears when invoked in a browser, see Figure 7–3 on page 7-4.

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 UI Servlet.

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 Oracle Configurator window can be further customized in Configurator Developer.

The Oracle Configurator window 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.

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.

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. The User Interface defined in Oracle Configurator Developer can include BMP files if it is only used with the Oracle Configurator in the Oracle SellingPoint application.

Customizing the HTML Templates

The template that you are most likely to modify is the Header Frame. For the HTML of this frame, see Example D–1, "The Header Frame template file (czHeader.htm)" on page D-1.

You may also want to modify the frameset files for the Netscape and Internet Explorer browsers (czFraNS.htm and czFraIE.htm, respectively). See Table 6–1 on page 6-3.

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.

Specifying the Location of Media Files

The HTML Template uses a global static variable IMAGESPATH to identify the location of media files, such as icons, backgrounds, button shapes, and the like. This variable is hardcoded by default to `/media/`. In your page designs, you should set only the file name of an image file, not the complete path to it. For instance,

```
var IMAGESPATH = '/media/';
```

in Example D–1, "The Header Frame template file (czHeader.htm)" on page D-1.

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 do so by setting a Java System Property Parameter to be passed to the UI 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 and SellingPoint Administration Guide*

Customizing 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 6-7.

Figure 6–2 The czHeader frame

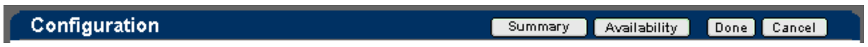


Table 6–3 Buttons in the Header Frame

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. This applies to both the Oracle Configurator UI and the Summary screen. When triggered for the Oracle Configurator UI, it adds the ATP information to the existing Model node controls. 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.

As a convenience, the complete source code for czHeader.htm is provided in Example D–1 on page D-1.

Customizing the Left and Right Border Frames

These frames load the HTML pages **czLeft.htm** and **czRight.htm**, respectively. They each display only a background color. You can add images and controls as required for your host application.

7 Testing

You can test the configuration window in these ways:

- If you are working in Oracle Configurator Developer (for instance, to develop your Model or User Interface, you can open a browser containing the configuration window directly from Configurator Developer. See Testing in Oracle Configurator Developer.
- If you are working outside Oracle Configurator Developer, you can open a browser containing the configuration window from an HTML test page. See Testing Outside Oracle Configurator Developer on page 7-6.

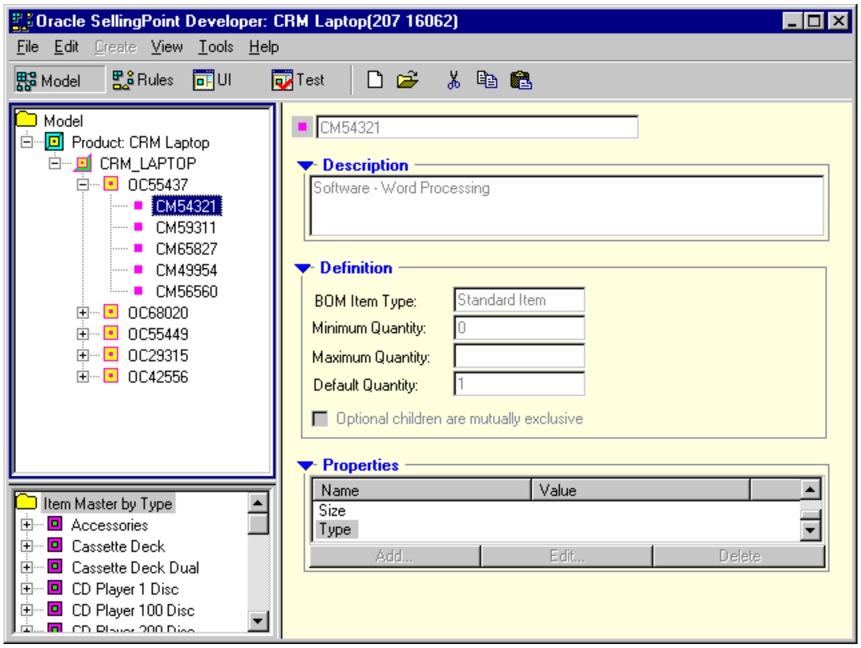
Testing in Oracle Configurator Developer

You can test the behavior of your Model and Configuration Rules, and the appearance of your User Interface, by using the Test/Debug module of Oracle Configurator Developer. Figure 7–1 shows a Model in Configurator Developer.

You can choose the environment in which to test:

Dynamic HTML in a browser	The Model and User Interface are displayed in your default web browser, in a frame built with the Toolkit HTML templates. See Figure 7–3.
Oracle SellingPoint application	The Model and User Interface are displayed in the Oracle SellingPoint application. See Figure 7–4.

Figure 7–1 A Model, in Configurator Developer


Oracle SellingPoint Developer: CRM Laptop(207 16062)
File Edit Create View Tools Help
Model
Rules
UI
Test
Add...
Edit...
Delete

Model
- Product: CRM Laptop
  - CRM_LAPTOP
    - OC55437
      - CM54321
      - CM59311
      - CM65827
      - CM49954
      - CM56560
    - OC68020
    - OC55449
    - OC29315
    - OC42556

Item Master by Type
- Accessories
- Cassette Deck
- Cassette Deck Dual
- CD Player 1 Disc
- CD Player 100 Disc
- CD Player 200 Disc

CM54321

Description

Software - Word Processing

Definition

BOM Item Type:	Standard Item
Minimum Quantity:	0
Maximum Quantity:	
Default Quantity:	1

Optional children are mutually exclusive

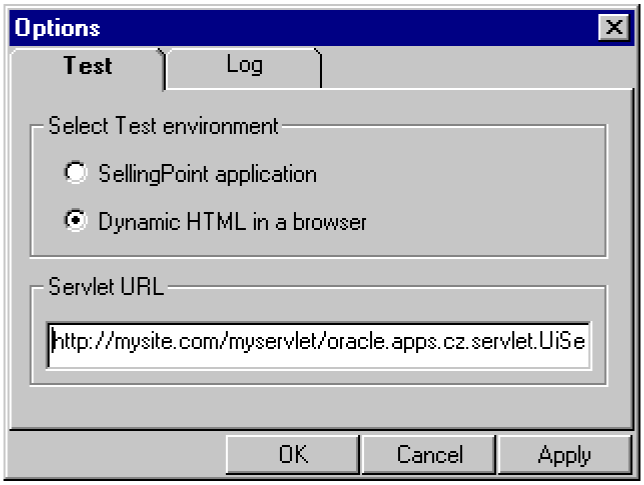
Properties

Name	Value
Size	
Type	

Testing DHTML in a Browser Window

1. Log in to Configurator Developer, using the same userid, password, and datasource name that you would use for your host application. Configurator Developer and the UI Servlet must access the same database.
2. In Configurator Developer, choose Tools > Options, then choose the Test tab in the Options dialog.

Figure 7–2 Test Tab in Options Dialog



SellingPoint application [✓ Dynamic HTML in a browser]

Servlet URL: http://mysite.com/myservlet/oracle.apps.cz.servlet.UiSe

3. Select the “Dynamic HTML in a browser” option.
4. In the Servlet URL field, enter the full URL of the UI Servlet for your host application.

   This URL is described in Chapter 2, "Installing the Servlet" and Chapter 3, "Integration: Session Initialization". It is of the form:

   ```
   http://host:port/servlet/oracle.apps.cz.servlet.UiServlet
   ```

   such as:

   ```
   http://www.mysite.com:60/myservlet/oracle.apps.cz.servlet.UiServlet
   ```

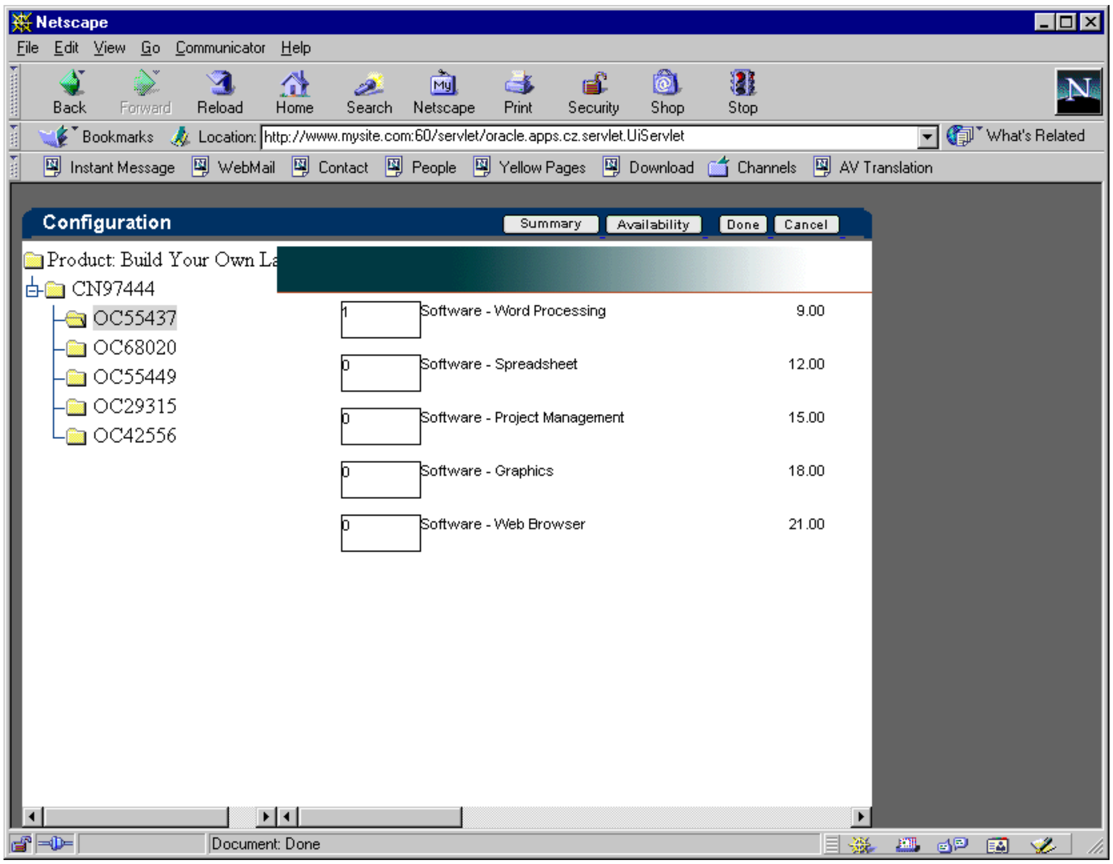
5. Click OK.
6. Click the Test button on the menu bar of Configurator Developer.
7. If you have defined more than one User Interface in Configurator Developer, a popup menu lists their names. Select the one that you want to test.
8. Your default web browser opens, displaying the current Model using the selected User Interface, in a frame built with the Toolkit HTML templates. See Figure 7–3 on page 7-4 for an example.

This frame is generated in the same way as the Toolkit configuration window in your host application. (Note: The UI elements shown in Figure 7–3 may differ slightly from the results of your test.)

Your current session information (database instance, username, password, etc.) is used to connect to the Oracle Configurator schema.

Configurator Developer has sent an initialization message to the UI Servlet, using the URL that you just specified. The User Interface definition is identified by the `ui_def_id` parameter. (See ui_def_id on page 3-20.)

Figure 7–3   A Model, in the OCT Configuration Window



Location: http://www.mysite.com:60/servlet/oracle.apps.cz.servlet.UiServlet

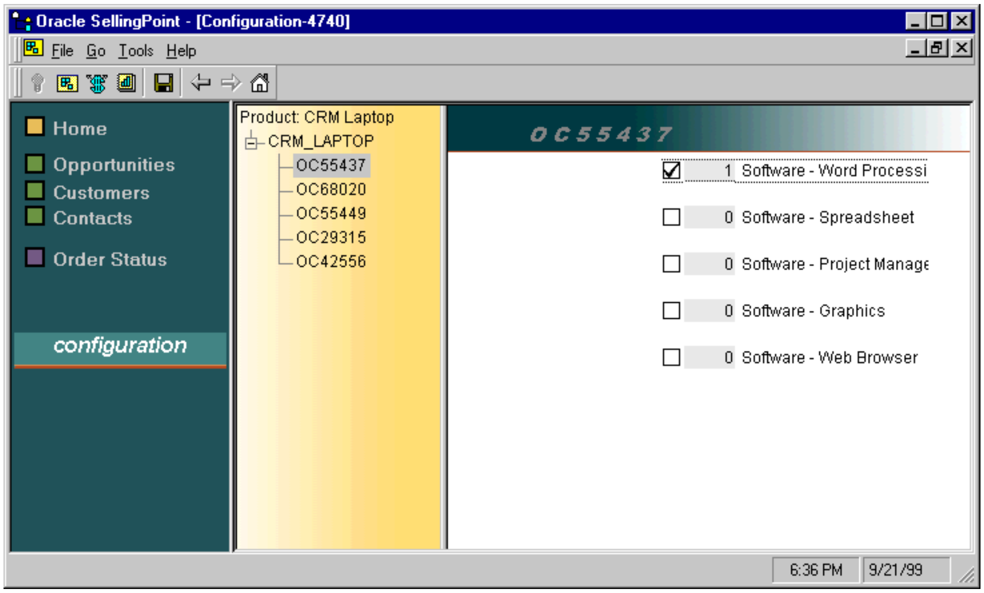
1	Software - Word Processing	9.00
0	Software - Spreadsheet	12.00
0	Software - Project Management	15.00
0	Software - Graphics	18.00
0	Software - Web Browser	21.00

Testing in the Oracle SellingPoint Application

1. Log in to Configurator Developer, using the same userid, password, and datasource name that you would use for your host application.
2. In Configurator Developer, choose Tools > Options, then choose the Test tab in the Options dialog. See Figure 7–2 on page 7-3
3. Select the "SellingPoint application" option.
4. Click OK.
5. Click the Test button on the menu bar of Configurator Developer.
6. The Oracle SellingPoint application runs, showing your Model and User Interface, as shown in Figure 7–4.

   See the *Oracle Configurator Developer User's Guide* and the *Oracle SellingPoint Application Help* for more information.

Figure 7–4   A Model, in the Oracle SellingPoint application



[✓] 1 Software - Word Processi
[ ] 0 Software - Spreadsheet
[ ] 0 Software - Project Manage
[ ] 0 Software - Graphics
[ ] 0 Software - Web Browser

Testing Outside Oracle Configurator Developer

You can test the configuration window outside Configurator Developer, by creating an HTML test page that substitutes for your host application.

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

   See Chapter 3, "Integration: Session Initialization" for an explanation of the OCT initialization message. See Example 3–3 for an example of a simple test page.

   For convenience, you can copy the initialization message produced by testing the configuration window from Configurator Developer. See Testing DHTML in a Browser Window on page 7-2 for information. After your browser opens, showing the configuration window, copy the URL that was inserted in the browser's location field, and edit it to fit the format of the initialization message.

2. Insure that you have the necessary database connectivity, and that your UI Servlet is installed and configured correctly. See Chapter 2, "Installing the Servlet".

3. Test the configuration window by opening the HTML test page. Your default web browser opens, displaying the current Model using the selected User Interface, in a frame built with the Toolkit HTML templates. See Figure 7–3 on page 7-4 for an example.

A

Troubleshooting Servlet Installation

This appendix provides suggestions for resolving problems that may arise when installing the Toolkit servlet. This installation procedure is described in Chapter 2, "Installing the Servlet" on page 2-1.

Some typical problems in setting up the servlet in Oracle Application Server involve:

- paths
- environment variables
- database connections

This appendix contains some basic tests that help you diagnose such problems.

Miscellaneous

- Make sure that you have set your virtual paths correctly. This is an issue if you are installing multiple OAS instances, or multiple cartridges. See Multiple Servlet Scenarios on page 2-19.
- Make sure that your executable path includes the Shared Object files, as listed in Files for the Servlet Directory on page 2-5, in Table 2–7 on page 2-5. A symptom of this problem might be an error message starting with a line like this one:

  ```
  java.lang.UnsatisfiedLinkError: no ocijdbc8 in shared library path
  ```

Checking the Operation of Oracle Application Server

What you are checking

Does Oracle Application Server work at all?

The test

Open this URL in a web browser:

`http://`*`yourserver`*`.com:`*`nnnn`*`/oasmgr`

where *`yourserver`* is the server you have just installed on, and *`nnnn`* is the port number configured for the OAS manager. Note that the port number for the OAS manager is usually different from and the port number for applications.

If the test fails

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

Checking the Response of the UI Servlet

What you are checking

Does the UiServlet respond to a test message?

The test

Open this URL in a web browser:

```
http://yourserver.com:mm/virtual_
path/oracle.apps.cz.servlet.UiServlet?test=test_string
```

where *`yourserver`* is the name of your server, *`mm`* is the port number for your host application, *`virtual_path`* is the virtual path that you set up when you installed the servlet cartridge, and *`test_string`* is an arbitrary unquoted character string. If the servlet is installed correctly, it should return an HTML page that simply prints *`test_string`*.

You can also obtain an HTML page giving the build version and expected schema, by entering `version` as the *`test_string`*.

If the test fails

If the test doesn’t work:

In OAS, turn on the maximum amount of logging, and set the Java System Property VERBOSE equal to TRUE in the Java Environment Parameters form (see VERBOSE on page 2-16).

Look in the log file produced by OAS to see which classes it loads, and from which JAR files. Towards the end of this file, there may be a message that some class failed to load. It is probably the case that there is a JAR file in the list that is not in the path specified or that there was an error in specifying its name.

B
The Checkout Servlet

Example B–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.

The Return URL	on page 4-10
Integration: Session Termination	on page 4-1
Submission	on page 4-3
Configuration Outputs	on page 4-5
<valid_configuration>	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 B–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 = \"/servlet/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 totalPrice = 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;
  }
}
```

C

Examples of Pricing and ATP Callback Procedures

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

See the following sections for background:

Integration: Pricing and ATP	on page 5-1
The Pricing Callback Interface	on page 5-5
The ATP Callback Interface	on page 5-7
Pricing Parameters	on page 3-10
ATP Parameters	on page 3-12

Example C–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 C–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;

  -- hard-coded price amount
  p_total_price := 343.00;
END price_multiple_items;
```

Example C–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;
```

D

The Header HTML Template

Example D–1 is the complete source code for the file **`czHeader.htm`**, which is provided with Toolkit as a template for constructing your own header frame for the configuration window.

See the following section for background.

Customizing the Header Frame on page 6-6

Example D–1 The Header Frame template file (czHeader.htm)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML DIR="LTR">
<head>
<!-- $Header: czHeader.htm 115.3 2000/03/20 20:02:46 snadkarn ship $ -->
<!--===================================================================+
 |     Copyright (c) 1999 Oracle Corporation, Redwood Shores, CA, USA      |
 |                       All rights reserved.                              |
 +=====================================================================+
 | HISTORY                                                                 |
 |                                                                         |
 |      xxxx-xx-99  K MacClay, CK Jeyaprakash Created.                     |
 |      xxxx-xx-99  K MacClay Added comments for end users.                |
 +===================================================================-->
<title></title>

<!-- Define Cascading Style Sheet (CSS) entries to enabled absolute     positioning of the
document elements.          (in order of style sheet appearance...)     - left-side tab (corner)
of the header background     - center tab stretched across frame on which the buttons are
rendered     - right-side tab or corner     - caption     - button layer which contains the
function-bound image layers-->

<style type="text/css">
```

```
  #DIV-tableft   {position:absolute ; left: 0;   top: 20; width: 7;   height: 21}
  #DIV-tabcenter {position:absolute ; left: 7;   top: 18; width: 624; height: 21}
  #DIV-tabright  {position:absolute ; left: 631; top: 20; width: 7;   height: 21}
  #DIV-label     {position:absolute ; left: 15;  top: 20; width: 23;  height: 21}
  #DIV-buttons   {position:absolute ; left: 360;  top: 23; width: 270; height: 21}
</style>

<script language="JavaScript">
<!--// begin browser no-script spoof

  /* Define global boolean values to signal browser type to other
     functions on this page. */
var ns4 = (document.layers)? true : false;
var ie4 = (document.all)? true : false;

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

  /* Time in milleseconds for the page to wait before resetting
     button images. */
  var ROLL_RESET_TIMEOUT = 2000;

  /* Enumeration of button image element indices...  these are
     needed to access the images through the Netscape DOM.

     Add more constants here according to the layout of your page. */
  var BTNIX_VIEW = 0;
  var BTNIX_ATP = 1;
  var BTNIX_DONE = 2;
  var BTNIX_CANCEL = 3;

  /* Flag to indicate that this docuement cannot be written to
     by the UiBroker's publish methods. */
var isRenderable = false;


  /* The following code hadles browser resize for Netscape browsers */
  function handleResize() {
    location.reload();
    return false;
  }

  if (ns4) {
    window.captureEvents(Event.RESIZE);
    window.onresize = handleResize;
  }

//--> end browser no-script spoof
</script>

<script language="JavaScript">
  var ub = null; // stores local reference to the uibroker object

  /** Funtion getUiBroker returns a pointer to the uibroker object
    * which is 'always' located in the czSrc frame. If this reference is
    * invalid due to customization,  this function must be modified.
    *
    * A null return value indicates that there was no instance to be found.
    * If this occurs a timeout and retry is recommended. It's likely that
    * on load, the browser has completed downloading this frame first.
    */
  function getUiBroker()
  {
    if (ub == null) {
      // make sure the czSrc frame has been loaded
      if (parent.frames.czSrc.isLoaded) {
        ub = parent.frames.czSrc.getUiBroker();
      }
    }
    if (ub == null) {
       //log that the UiBroker has not been found. Some sort of internal error!!
    }
    return ub;
  }

  /** Function viewClick handles a 'mouseclick' on the view button.
    * By default this toggles between the configuration and summary
    * screens by requesting that the server publish to the content
    * frame
    */
  function viewClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      //setting the reference of this page in uibroker for futher reference
      uBroker.hdr_ref = this;

      // check the current state of the content frame
      // through the uibroker's method (function).
      if (uBroker.getContentMode() == 'config') {
        // ask the server to publish the summary screen.
    uBroker.showSummary();
        setImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czConf0.gif');
        // roll the current image
        rollImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czConf1.gif', IMAGESPATH +
'czConf0.gif');
      }
      else {
        // ask the server to publish the configuration screen.
                         if (uBroker.pageHistory.peek () ==  '<show-summary/>')
                         {
                            uBroker.pageHistory.pop();
                         }
 uBroker.showConfiguration();
        setImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czSumm0.gif');
        // roll the current image
        rollImage('IMG-view', BTNIX_VIEW, IMAGESPATH + 'czSumm1.gif', IMAGESPATH +
'czSumm0.gif');
}
    }
    else {
      // wait for the uibroker.
    }
  }

  /** Function doneClick handles a 'mouseclick' on the done button.
    * By default this initiates a uibroker function call,  which, in turn,
    * communicates to the server that the configuration session is complete.
    *
    * Code may be added here to signal a containing frame that the configuration
    * is complete.
    */
  function doneClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      // tell the server we are done.
  uBroker.completeConfiguration();
      rollImage('IMG-done', BTNIX_DONE, IMAGESPATH + 'czDone1.gif', IMAGESPATH + 'czDone0.gif');
    } else {
      // wait for the uibroker.
    }
  }

  /** Function cancelClick handles a 'mouseclick' on the cancel button.
    * By default this initiates a uibroker function call,  which, in turn,
    * communicates to the server that the configuration session is canceled.
    *
    * Code may be added here to signal a containing frame that the configuration
    * is complete.
    */
  function cancelClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      // tell the server we are done.
  uBroker.cancelConfiguration();
      rollImage('IMG-cancel', BTNIX_CANCEL, IMAGESPATH + 'czCanc1.gif', IMAGESPATH +
'czCanc0.gif');
    } else {
      // wait for the uibroker.
    }
  }

  /** Function atpClick handles a 'mouseclick' on the atp button.
    * By default this initiates a uibroker function call,  which, in turn,
    * communicates to the server that Availability to Promise should be displayed
    * with items in the interactive controls (lists/checkboxes/etc.) located in the
    * czDisp - display frame.
    *
    * Code may be added here to signal a containing frame that the ATP is being
    * displayed.
    */
  function atpClick()
  {
    var uBroker = this.getUiBroker();

    if (uBroker != null) {
      // tell the server we are done.
  uBroker.showATP();
      rollImage('IMG-atp', BTNIX_ATP, IMAGESPATH + 'czAvail1.gif', IMAGESPATH + 'czAvail0.gif');
    } else {
      // wait for the uibroker.
    }
  }
  /** Function setImage changes the src attribute of an image on the buttons layer.
    * NOTE: Netscape requires the element index to grab the image appropriately.
    */
  function setImage(imgId, elementIndex, newSrc) {
var block = null;
var divId = 'DIV-buttons';

    if (ns4) {
  block = document.layers[divId];
  block.document.images[elementIndex].src = newSrc;
  } else if (ie4) {
    block = document.all[divId];
  block.all[imgId].src = newSrc;
  }
  }

  /** Function rollImage swaps images to give the buttons a clicked affect.
    * The original image is generally restored with a timeout based on a page level
    * constant.
    *
    * NOTE: Netscape requires the element index to grab the image appropriately.
    */
  function rollImage(id, elementIndex, newSrc, oldSrc)
  {
    setImage(id, elementIndex, newSrc);
    if (oldSrc) {
      resetImageAfterMilliseconds(id, elementIndex, oldSrc, ROLL_RESET_TIMEOUT);
    }
  }

  /** Function resetImageAfterMilliseconds restores images to return buttons
    * to their original images.
    * The original image is gnerally restored with a timeout based on a page level
    * constant.
    */
  function resetImageAfterMilliseconds(id, elementNum, oldSrc, ms)
  {
    var funcStr = "setImage('" + id + "', " + elementNum + ", '" + oldSrc + "')";
    setTimeout (funcStr, ms);
  }

//--> end browser no-script spoof
</script>
</head>
<body bgcolor="#666666">
```

```
  <!-- background images -->
    <!-- left tab -->
    <div id="DIV-tableft" nowrap>
      <img src="/media/czTabL.gif" width="7" height="21">
    </div>

    <!-- center tab -->
  <div id="DIV-tabcenter" nowrap>
      <img src="/media/czTabC.gif" width="624" height="21">
    </div>

    <!-- right tab -->
    <div id="DIV-tabright" nowrap>
      <img src="/media/czTabR.gif" width="7" height="21">
    </div>
  <!-- end background images -->

  <!-- label -->
<div id="DIV-label" nowrap>
    <font face="Arial" pointsize="12pt" color="white">
      <b>Configuration</b>
    </font>
  </div>

  <!-- buttons (anchor/image) layer -->
<div id="DIV-buttons" nowrap>

    <!-- view toggle (Configuration/Summary screens) -->
    <a href="javascript:viewClick();">
      <img name="IMG-view" src="/media/czSumm0.gif" border="0" WIDTH="73" HEIGHT="15">
    </a>

    <!-- show Availability to Promise (ATP) button -->
    <a href="javascript:atpClick();">
      <img name="IMG-atp" src="/media/czAvail0.gif" border="0" WIDTH="73" HEIGHT="15">
    </a>

    <!-- add some space between the buttons. -->
    &nbsp;

    <!-- done button -->
    <a href="javascript:doneClick();">
      <img name="IMG-done" src="/media/czDone0.gif" border="0" WIDTH="37" HEIGHT="15">
    </a>
    <!-- cancel button -->
    <a href="javascript:cancelClick();">
      <img name="IMG-cancel" src="/media/czCanc0.gif" border="0" WIDTH="50" HEIGHT="15">
    </a>

  </div>

</body>
</html>
```

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.

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 Material

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

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

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.

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

Data Integration Object. 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.

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.

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, Oracle Configurator window, 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.

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 Module

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