Oracle® Configurator

Performance Guide

Release 11*i*

Part No. B13608-01

December 2005

This book explains the fundamentals of performance tuning and provides detailed information about setting up the Web server for optimal Oracle Configurator runtime performance. It also provides Configurator-specific setup parameters for using LoadRunner, a useful tool for load testing and collecting performance data.

ORACLE®

Oracle Configurator Performance Guide, Release 11*i*

Part No. B13608-01

Primary Author: Stephen Damiani

Contributing Author: Tina Brand

Contributor: William Brand, George Colpitts, Ivan Lazarov, Paul Perrone, Martin Plotkin, Anil Tandon

Contents

3 Web Server Configuration

List of Examples

List of Figures

List of Tables

Send Us Your Comments

Oracle Configurator Performance Guide, Release 11*i*

Part No. B13608-01

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

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

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

- Electronic mail: czdoc_us@oracle.com
- FAX: 781-238-9898. Attn: Oracle Configurator Documentation
- Postal service:

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

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

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

Preface

Welcome to the *Oracle Configurator Performance Guide*. This book contains information you need for optimizing runtime performance of Oracle Configurator. Using this document, you can prepare for and implement high performance configuration model designs, Web server configuration, and runtime testing. This book should not be used alone, but together with the other books in the Oracle Configurator documentation set.

This preface describes how the book is organized, who the intended audience is, and how to interpret the typographic conventions.

For details about what is and is not included in this book, see Section 1.1, "Scope of this Guide" on page 1-1.

Intended Audience

This guide is intended for Oracle Consultants and implementers who are deploying a Web-based Configurator. You should have a working knowledge of your business processes, tools, configuration models, and the other books in the Oracle Configurator documentation set. You should also be familiar with Oracle Applications and the Oracle Applications database.

If you have never used a configurator application, we suggest you attend one or more of the Oracle Configurator training classes available through Oracle University.

Additional specifics about audience are included in the Structure section of this preface.

Documentation Accessibility

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

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

Accessibility of Code Examples in Documentation Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some

screen readers may not always read a line of text that consists solely of a bracket or brace.

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

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

Structure

This book contains a table of contents, lists of examples, tables, and figures, a reader comment form, a preface, several chapters, appendixes, a glossary, and an index.

Introduction

Useful for anyone planning, implementing, or testing Oracle Configurator, this chapter explains the scope of the guide, defines terms, and provides background information about Oracle Configurator that is relevant to evaluating and tuning runtime performance.

Tuning

Useful for anyone preparing Oracle Configurator for deployment, this chapter describes the tuning process and best practices for tuning performance over the lifetime of your project, as well as important tuning recommendations not included in the *Oracle Configurator Modeling Guide* or Chapter 3, "Web Server Configuration".

Web Server Configuration

Intended principally as a reference for the Web server administrator or the database administrator responsible for configuring and maintaining Web servers, this chapter lists default and recommended values for parameters of the Web server that affect Oracle Configurator performance.

Tools and Collecting Data

Intended principally as a reference for anyone involved in testing Oracle Configurator, this chapter explains what and under what conditions to collect data for performance evaluation and tuning. LoadRunner (Mercury Interactive Corporation) is the sample tool. Appendix B, "LoadRunner Settings" lists specific LoadRunner settings

Summary of Web Configuration Parameters

Intended for the Web server administrator or the database administrator responsible for configuring and maintaining Web servers, as an overview at a glance of the Apache Web server parameters that affect Oracle Configurator, their recommended values, and the file in which the parameter is set.

LoadRunner Settings

Intended principally as a reference for anyone setting up LoadRunner tests, this appendix describes some LoadRunner parameters that are specific to Oracle Configurator and that must be set in order to collect performance data.

DHTML User Interfaces

Intended as a reference only for installations using legacy DHMTL User Interfaces, this appendix provides suggestions for tuning the network and a client machine for optimal performance.

Glossary

This section is a glossary of terms used throughout the Oracle Configurator documentation set.

Related Documents

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

- *Oracle Configurator Implementation Guide* (Part No. B13604-03)
- *Oracle Configurator Installation Guide* (Part No. B13602-03)
- *Oracle Configurator Developer User's Guide* (Part No. B13603-03)
- *Oracle Configurator Extensions and Interface Object Developer's Guide* (Part No. B13607-03)

The following documents may also be useful:

- *Oracle Configurator Release Notes* (Part No. B10620-01)
- *Oracle Applications System Administrator's Guide (Part No. A75396–08)*
- *LoadRunner: Creating GUI Vuser Scripts Online Guide*
- *Oracle Applications User's Guide (Part No. A75394–04)*
- *Oracle Applications Developer's Guide (Part No. A75545–03)*
- *Maintaining Oracle Applications* (Part No. A90810–01)
- Oracle 9i Database Performance Methods (Part No. A87504-02)
- Oracle 9i Database Performance Planning (Part No. A96532-01)
- Oracle 9i Database Performance Tuning Guide and Reference (Part No. A96533-01)

The following documents represent a possible starting point in researching tuning and benchmarking methodology:

- *Quality Web Systems: Performance, Security, and Usability* by Elfriede Dustin, Jeff Rashka, Douglas McDiarmid, Jakob Nielson (Addison-Wesley Longman, 2001, ISBN: 0201719363)
- *Performance Evaluation and Benchmarking with Realistic Applications*, Rudolf Eigenmann, Editor (MIT Press, 2001, ISBN 0262050668)

Additional white papers and information about benchmarking and tuning are available at:

- http://www.benchmarkresources.com
- http://www.ideasinternational.com/benchmark/bench.html
- http://www.oracle.com/apps_benchmark/html/index.html?results.html

Other sites of interest are:

- http://httpd.apache.org/docs-project/

- http://apache.us.oracle.com

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
boldface text	Boldface type in text indicates a new term, a term defined in the glossary, specific keys, and labels of user interface objects. Boldface type also indicates a menu, command, or option, especially within procedures
italics	Italic type in text, tables, or code examples indicates user-supplied text. Replace these placeholders with a specific value or string.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
>	The left bracket alone represents the MS DOS prompt.
$	The dollar sign represents the DIGITAL Command Language prompt in Windows and the Bourne shell prompt in Digital UNIX.
%	The percent sign alone represents the UNIX prompt.
`name()`	In text other than code examples, the names of programming language methods and functions are shown with trailing parentheses. The parentheses are always shown as empty. For the actual argument or parameter list, see the reference documentation. This convention is *not* used in code examples.

Product Support

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

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

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

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

1. Choose the **TARs** link in the left menu.
2. Click on **Create a TAR**.
3. Fill in or choose a profile.
4. In the same form:
   a. Choose **Product**: Oracle Configurator or Oracle Configurator Developer

b. Choose **Type of Problem**: Oracle Configurator Generic Issue template

5. Provide the information requested in the iTAR template.

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

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

`www.oracle.com/support/`

Troubleshooting

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

For information about logging while working in Configurator Developer, see the *Oracle Configurator Developer User's Guide*.

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

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

1 Introduction

This Introduction explains the scope of the *Oracle Configurator Performance Guide* and relevant terms used in the book.

For an overview of Oracle Configurator architecture, see the *Oracle Configurator Implementation Guide*.

1.1 Scope of this Guide

This guide focuses primarily on the Applications tier of Oracle Configurator's architecture. See Section 2.3, "Tuning Key Components of the Oracle Configurator Architecture" on page 2-4 for details about what this tier includes.

Within the Applications tier, this guide focuses on the issues involved in accurately measuring performance and scalability, as well as the analysis tools used to measure performance. The guide discusses how to plan and conduct a comprehensive benchmark for an Oracle Configurator implementation.

The main areas of potential performance gain are those where most of the runtime processing takes place:

- Configuration Model Design
- Web Server Configuration

This guide focuses on information specific to Oracle Configurator in these areas, but also provides explanations of tuning potential in other Oracle Configurator components of a deployment, such as using Oracle Configurator Developer. For details, see Chapter 2, "Tuning".

The following topics are also important for tuning performance, but are *not* the focus of this book:

- Tuning the database or database server (some suggestions are presented in Chapter 2, "Tuning")
- Determining machine sizes for your deployed application
- Tuning non-runtime performance such as Oracle Configurator Developer, configuration model publication, or data import and refresh performance
- Tuning the host application
- Tuning the client machine setup (This applies only if you are using legacy DHTML User Interfaces. See Appendix C, "DHTML User Interfaces" for some suggestions.)

1.2 Factors that Influence Runtime Performance

Oracle Configurator is a memory and Central Processing Unit (CPU)-intensive application. As such, following are the primary factors that influence runtime performance:

- Speed and number of CPUs
- Number of concurrent users per CPU
- Number and size of configuration models
- Number of users per Java Virtual Machine (JVM)
- Total number of JVMs
- Number of JVMs per CPU
- Number of CPUs per server
- Amount of memory per server
- Oracle *i*AS and Apache configuration settings

For more information about each of these factors, see Chapter 2, "Tuning" and Chapter 3, "Web Server Configuration".

1.2.1 Architecture

The runtime Oracle Configurator, Oracle Configurator Developer and CZ schema run as part of the Oracle Applications eBusiness Suite in a multi-tier architecture. The runtime Oracle Configurator runs in a three-tier architecture. These tiers include:

- The Client tier
- The Middle tier
- The Data tier

Each tier is described in more detail in the *Oracle Configurator Implementation Guide*.

1.2.2 Configuration Model Design

The design of the configuration model is critically important to the performance of your runtime Configurator. Good design that considers performance implications can reduce the need for additional changes to the configuration model during performance testing. Careful design and tuning of the configuration model, with special emphasis on the size and complexity of the Model, usually yields large performance dividends.

For more information, see the *Oracle Configurator Modeling Guide*.

1.2.3 Web Server Configuration

A properly configured Web server greatly contributes to the performance of the runtime Oracle Configurator. There are several configuration parameters in the Apache Web server and the JServ module which must be tuned to provide optimal performance. For details, see Chapter 3, "Web Server Configuration" on page 3-1.

2

Tuning

This chapter explains the fundamentals of performance tuning and how various components of the Oracle Configurator architecture affect performance.

> **Note:** The tuning process should begin during the configuration model design phase. Modifying the design to improve performance after the configuration model has been built may significantly delay deployment. For details, see the *Oracle Configurator Modeling Guide*.

This chapter contains the following sections:

- Fundamentals of Performance Tuning
- The Tuning Process
- Tuning Key Components of the Oracle Configurator Architecture
- Isolating Performance Problems and Intervening

See the "Structure" section in the Preface to learn about the audience for which this chapter is intended.

2.1 Fundamentals of Performance Tuning

Performance tuning is a complex, iterative process. Deriving meaningful results requires using a proven methodology and understanding the environment, how the application will be used, and the required throughput. Using a proven methodology minimizes confusion and gaps in the process. By understanding the environment, application, and throughput, you will be able to define the kind of benchmarks you need to accurately reflect the scenario in which you operate.

> **WARNING: The performance tuning methodology presented here introduces the topic and is not meant to be exhaustive or definitive. Consult the "Related Documents" section in the Preface of this book for help in researching a tuning methodology for your implementation.**

2.1.1 Methodology

A methodology provides a general approach to performance tuning that helps you avoid gaps or missed opportunities. At the core of a useful methodology is a repeatable process for evaluating, diagnosing, and intervening to improve performance.

In the case of Oracle Configurator, intervention can consist of:

- Modifying the configuration model design
- Changing the Web server configuration
- Deploying a Java Servlet Engine (JServ) that is dedicated to Oracle Configurator (in other words, only Oracle Configurator runs on a specific JServ, rather than multiple applications).

  For details, go to Metalink, Oracle's technical support Web site at:

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

- Upgrading to a more recent release
- Applying available patches
- Modifying your hardware setup

Additional intervention can occur in the database and integration points with hosting applications.

Depending on your needs, intervention should generally maximize the largest potential benefit for the broadest range of circumstances.

Comprehensive documentation of baselines and benchmarks is a prerequisite to effective analysis and comparisons. See Section 2.1.2, "Documenting Results" on page 2-2.

Successful diagnosis and tuning requires:

- Understanding your expectations of success and determining what are reasonable and achievable goals
- Dividing the application's key components into measurable units and evaluating their performance separately, as well as collectively
- Having access to expert knowledge of your business model, the application, the workflow of operation, and the environment and technology stack
- Having access to expert knowledge of Oracle Configurator, including Oracle Configurator Developer, for understanding the configuration model's role in performance
- Benchmarking the application at various usage levels
- Repeating the tuning process when measurements fall short
- Recording steps and documenting results to make the process reliable and repeatable

For more information about collecting and analyzing data, see Chapter 4, "Tools and Collecting Data".

2.1.2 Documenting Results

It is important to maintain detailed notes during each phase of the data collection, analysis, and tuning process. Detailed documentation of setup, actions, and results helps ensure that the process is reproducible and reliable. Evaluating or analyzing the problem includes understanding what has been done to date and comparing performance with baselines and previous benchmarks.

2.1.3 Defining Success Criteria

You will need to determine at what point in the tuning process performance of your configuration model is acceptable for deployment. This requires balancing what can be expected from your software and hardware with what your end users find acceptable. Conduct usability studies with sample end users to determine realistic performance goals.

2.2 The Tuning Process

By following a repeatable and reliable methodology, the tuning process allows you to make decisions and changes based on appropriate analysis of the data. Tuning Oracle Configurator effectively requires following a proven, iterative sequence of tasks and specific steps within those tasks.

2.2.1 Steps in the Process

Tuning Oracle Configurator is a step-by-step process.

1. Collect performance data for various operations within Oracle Configurator using the tools described in Chapter 4, "Tools and Collecting Data".

   For a list of runtime events and other functionality where performance problems may occur, see Section 2.4, "Isolating Performance Problems and Intervening" on page 2-7.

2. Document the results. If the performance numbers are not satisfactory, begin tuning.

   If the Oracle Configurator implementation has never been performance tested or tuned before, continue with the process described here.

   If the Oracle Configurator implementation has previously performed better, either repeat this process or see the suggestions in Section 2.4.5, "Tuning Hot Spots" on page 2-9.

3. Tune the configuration model design.

   The sequence of tuning tasks within the configuration model is as follows:

   a. Model Structure

   b. User Interface

   c. Rules

   d. Configurator Extensions

   For details about each tuning these parts of your configuration model, see the *Oracle Configurator Modeling Guide*.

4. Collect incremental performance numbers after making a few changes. Document the changes made and the resulting performance numbers.

5. Iterate over the previous two steps until either the performance numbers are satisfactory or no more changes are possible to the Model structure, rules, User Interface, or Configurator Extensions.

6. Focus next on the Web server configuration. Web server tuning is necessary when performance for a single user is good, but degrades significantly with multiple users.

Monitor CPU utilization and memory usage for the most costly events, which include:

a. Model load

b. Configuration save

c. Configurator Extensions

7. Start the iterative process of tuning the Web server configuration.

   Following is the sequence of tasks for tuning the Web server configuration:

   a. Change configuration parameters as described in Chapter 3, "Web Server Configuration".

   b. Modify the number of Java Virtual Machines (JVMs) per CPU.

   c. Modify the maximum number of end users per JVM to avoid 100% CPU utilization and prevent the JVM from failing.

   d. Deploy a JServ consisting of one or more load balanced JVMs that is used only by Oracle Configurator.

      For details, go to Metalink, Oracle's technical support Web site at:

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

8. Document the changes made to the Web server and collect the resulting performance numbers.

9. Iterate over the previous two steps until either the performance numbers are satisfactory or no more changes are possible.

10. To improve performance when loading or preloading configuration models and saving configurations, Oracle recommends tuning the database. Database tuning is not within the scope of this book, but a few suggestions are provided in Section 2.3.1.5, "Database Server" on page 2-6.

It is important to maintain detailed notes during each phase of the tuning process. This makes the process reproducible and reliable.

2.3 Tuning Key Components of the Oracle Configurator Architecture

Oracle Configurator operates in a classic three-tier environment. The configuration model, user interface definitions and business rules are *stored* in the Data tier. These definitions are *processed* in the Applications tier and *rendered* in the Client tier (the User Interface). Most of the processing for Oracle Configurator is performed in the middle, or Applications, tier. In that sense, Oracle Configurator is a typical middle-tier application.

See the *Oracle Configurator Implementation Guide* for details about the architecture and application flow among tiers and elements of an Oracle Configurator deployment.

For best performance, set up your Oracle Configurator deployment with different server machines for the database and the application or Web server (middle tier). For details about this, see Section 2.3.2, "Server Machine Tuning" on page 2-6. Size the middle tier (where the OC Servlet with the Oracle Configurator engine and UI Server are installed) according to the complexity and size of your configuration model and end-user demands on your runtime Oracle Configurator.

Oracle recommends that you dedicate at least one JServ to running only Oracle Configurator. For details, go to Metalink, Oracle's technical support Web site at:

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

2.3.1 Components Affecting Performance

The optimal performance of a Web-based Oracle Configurator depends on the performance of the following elements.

- Configuration Model Design
- Configurator Extension Design and Implementation
- Custom User Interface Design and Implementation
- Web Server
- Database Server
- The Network

These components require careful analysis to get the best possible performance out of the system. Tuning requires iterating over all these essential components and tuning each of them both independently and collectively.

2.3.1.1 Configuration Model Design

All parts of a configuration model participate in performance. You benefit from designing each part thoroughly to optimize performance before starting model construction. For a review of what a configuration model is, see the Glossary. For details about tuning configuration model design, see the *Oracle Configurator Modeling Guide*.

2.3.1.2 Configurator Extension Design and Implementation

The CIO is an API layer that handles communication between the Oracle Configurator engine and the UI. Configurator Extensions and custom UIs communicate with the engine through the CIO.

There is generally no tuning that can be done in the CIO interactions. However, the design and implementation of code that uses CIO methods to access the configuration model (such as Configurator Extensions) can be tuned to use the CIO most effectively.

Configurator Extensions and custom UI code can affect runtime performance by less than optimal use of the CIO methods. You can optimize (or minimize) the sequence of changes to a configuration model that a Configurator Extension invokes by avoiding expensive intermediary transactions in the CIO.

For details on tuning Configurator Extensions, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

2.3.1.3 Custom User Interface Design and Implementation

In some deployments, the user interface is entirely custom, meaning it is not generated in Oracle Configurator Developer. Custom UI code must interface with the Configuration Interface Object (CIO), so it is important to optimize how the UI code calls the CIO. For more information, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

2.3.1.4 Web Server

Most of Oracle Configurator processing occurs in the Web server portion of the middle tier. Data retrieved from the database is processed in memory in the Web server.

Oracle Configurator uses the high performance Apache Web server in the Oracle Internet Application Server (*i*AS). Web server configuration file parameters that are used to tune performance are described in Section 3.2, "Apache Web Server Parameters" on page 3-1.

Perhaps the most important consideration affecting performance on the Web tier is how much memory is allocated to and used by the runtime Oracle Configurator. For example, consider the total number of Oracle Configurator users per JVM and the number and size of Models you can expect to be stored in the cache at one time. Also, determine the maximum heap size allocation for your platform and the optimal number of JVMs per CPU (the latter can be determined only by trial-and-error experimentation).

For more information, see Section 3.2.2.1, "Heap Size" on page 3-3.

You may also want to consider using a third-party application such as Optimizeit™ to identify and isolate Java-related performance issues.

2.3.1.5 Database Server

Oracle Configurator in the middle tier accesses the database when starting a configuration session and saving a configuration. Data is retrieved from the database in intervals and processed in memory in the Web server. See Section 2.4.1, "System Transactions" on page 2-7 for data transaction scheduling.

For more efficient use of database server resources, purge records flagged for deletion before creating the production-ready version. See the *Oracle Configurator Implementation Guide* for more information about purging records.

For important recommendation regarding the database server and Web server, refer to the Note in Section 2.3.2, "Server Machine Tuning" on page 2-6.

While there is no specific database tuning required for Oracle Configurator, following the general recommendations for tuning Oracle Applications is likely to improve performance.

Note: Tune the configuration model design first and then the Web server configuration. Finally, tune the database.

2.3.1.6 The Network

Network latency may affect production end users. Network latency may not be accounted for in benchmarking test results. Oracle recommends using a tool that allows you to isolate problems and improve performance when displaying data in a Web browser (for example, the freeware application called IBM Page Detailer).

The Web server and database servers should be on separate machines, but on the same high bandwidth network. Timeout settings of any firewall should be set to `none` to prevent the database connection from failing.

See the *Oracle Configurator Implementation Guide* for additional details about network considerations in an Oracle Configurator deployment.

2.3.2 Server Machine Tuning

Because Oracle Configurator processes compete with database processes for CPU time and OC Servlet transactions are longer than typical database transactions, the Web server and database server are never installed on the same machine. See Section 2.4.1,

"System Transactions" on page 2-7 for more information about how Oracle Configurator and database processes compete.

> **Note:** Oracle strongly recommends that the Web server, the database server, and the client all be installed on separate machines, but on the same local network.

Garbage collection (GC) is an important part of tuning a Java Web server application. Be sure to increase the memory as you increase the number of processors, since allocation can be parallelized, but GC is not parallel.

If server machine memory is an issue, a router may be needed to spread the end users to particular CPUs dedicated to supporting a subset of the possible configuration models. Routing end user access to specific servers running a subset of models requires additional planning and testing. See the *Oracle Configurator Implementation Guide* for additional details about using routers in an Oracle Configurator deployment.

2.3.3 Supported Browsers

For a list of supported browsers that all Oracle Applications Framework-based products support, see the Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

2.4 Isolating Performance Problems and Intervening

Optimizing the following can lead to significant performance gains the next time you test the system or run a benchmark:

- System Transactions
- Pricing and ATP
- Database Commits
- Oracle Applications Debugging and Performance
- Tuning Hot Spots

Beyond these common problem areas, you can explore and tune hot spots, as presented in Section 2.4.5 on page 2-9.

2.4.1 System Transactions

Runtime performance involves distinct events:

- Transmission of user action
- Processing of UI event by the Oracle Configurator Servlet
- Processing of logic changes by the Oracle Configurator engine
- Creation of UI changes by Oracle Configurator Servlet
- Transmission to and rendering by the client browser

Commonly, most of the processing occurs in the client to Web server network and UI rendering. Logic transactions are *relatively* faster when measured against that network and UI processing. However, rule design and the number of rules in a configuration model do affect performance of logic transactions in the OC Servlet. For details about designing rules for best performance, see the *Oracle Configurator Modeling Guide*.

Oracle Configurator is a CPU-intensive application. Data is retrieved from the database in intervals and processed in memory in the Web server. The default UNIX "round robin" CPU scheduling favors short transactions. Typical database transactions are short transactions. Typical Oracle Configurator Servlet transactions are long-running transactions that compete for CPU time.

> **Note:** In a busy system where CPU utilization is high, database processes take precedence over Oracle Configurator processes, and Oracle Configurator processes get only a small portion of the CPU at a time. This significantly affects performance (that is, server response time and display time). For this reason, running the database and Web server on separate machines in a distributed computing environment is critically important. See also Section 2.3.2, "Server Machine Tuning" on page 2-6.

2.4.2 Pricing and ATP

Runtime performance problems could arise from pricing and Available to Promise (ATP) processing within Oracle Configurator. You can improve performance by fetching pricing data only when it is required by your end users.

If displaying prices and ATP data is not required, isolate this process as a performance problem by disabling pricing and ATP. See Section 3.2.2.4, "`cz.activemodel`" on page 3-5 for details.

If displaying prices or ATP data is required, intervene by tuning the frequency of transactions involving retrieval of this information. Perform the following in Configurator Developer to minimize the performance impact:

- Display either list prices or selling prices (but not both)
- Set Recalculate Prices to On Request

You can modify these settings when editing the UI Definition of a User Interface that was created in Configurator Developer. For details, see the *Oracle Configurator Developer User's Guide*.

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

2.4.3 Database Commits

In some cases, hosting applications must commit end-user changes before launching Oracle Configurator. Initializing and loading the configuration model can be slow if an implicit save occurs. If the end user explicitly saves work in the host application before starting a configuration session, the perceived performance of the Oracle Configurator improves. Order Management is an example where an explicit save before launching Oracle Configurator significantly improves configuration model load time.

Saving data within a host application is not related to performance of the runtime Oracle Configurator. Therefore, be sure to save data in the host application before launching Oracle Configurator and recording performance statistics.

2.4.4 Oracle Applications Debugging and Performance

To improve performance of Configurator Developer and the runtime Oracle Configurator, be sure that any profile options and Java system parameters that are related to logging are disabled unless you are currently debugging an issue. Logging is

not enabled by default and should only be used to identify and resolve performance issues.

For more information, see "Troubleshooting" on page -xv.

2.4.5 Tuning Hot Spots

Ideally, you tune Oracle Configurator following the overall sequence outlined in Section 2.2.1, "Steps in the Process". When tuning for hot spots, the sequence may be different, particularly if limited changes have occurred in a runtime Oracle Configurator that previously performed well.

Hot spot tuning involves identifying the area of change in the implementation since the previous performance test, and tuning that area if the results show degradation. It is critical to have baseline performance data with which to compare the changes. In Section 2.4.5.1 through Section 2.4.5.5, the assumption is that only one factor in the implementation has changed. For example, if your configuration model has changed and nothing else has, what could be affecting your performance results?

2.4.5.1 If Your Configuration Model Has Changed

If your configuration model has changed, performance problems could manifest themselves in the following areas:

- Model load time, if the number of nodes has increased significantly (see the *Oracle Configurator Modeling Guide* for suggestions)
- Web server performance, if the rules or UI customizations have changed

Reducing the number of nodes in a configuration model is not usually an option. However, you can optimize configuration model load time by changing the way a large model is loaded, or by preloading it. See Section 3.2.4, "Preloading the Oracle Configurator Servlet" on page 3-6.

If rule and UI customizations follow the design suggestions in the *Oracle Configurator Modeling Guide* and there are no further design optimizations possible, you can improve Web server performance by adjusting parameters in the Web server configuration. For example, you may need to adjust the `Timeout` parameter. See Chapter 3, "Web Server Configuration" for details.

2.4.5.2 If Your Web Server Setup or Configuration Has Changed

Document any changes made to the Web server so you can evaluate why performance may have degraded and you can go back to a configuration that performed better. For example, routine maintenance or installation of a database or additional software could adversely affect performance. Changing one OC Servlet parameter to resolve a new issue may have runtime implications that require other parameters to be adjusted as well in order to sustain an acceptable level of performance. For example, the OC Servlet parameter `cz.activemodel` is modified and Oracle Configurator now retrieves and displays both pricing and ATP data. This type of small change can significantly affect performance. See Section 3.2, "Apache Web Server Parameters" on page 3-1.

Hardware issues could also be the cause of performance degradation. For example, one of the CPUs has failed and needs to be repaired or replaced.

2.4.5.3 If You Upgraded Oracle Applications

After an upgrade of your Oracle Applications release (for example from 11.5.9 to 11.5.10), identify what now performs more slowly. If there were no changes in the

Oracle Configurator setup, but runtime performance has degraded, a change may have been made in the host application.

For example, if it takes longer to save a configuration (performance after clicking the Done button in Oracle Configurator), then upgrading Order Management may have introduced a change that needs to be examined. Or if you have pricing enabled and you now see slower rendering of Oracle Configurator pages, the Advanced Pricing upgrade may have introduced a change that needs to be examined. In this case, you can easily isolate pricing as the problem by disabling it temporarily. See Section 2.4.2, "Pricing and ATP" on page 2-8.

2.4.5.4 If You Applied Oracle Configurator Patches

If no other changes have occurred, applying a patch or upgrading to the latest Oracle Configurator release may introduce functionality that affects performance.

> **Note:** *Before* applying an Oracle Configurator patch, read the ARU Patch Readme and the Oracle Configurator About document (Metalink Note 264934.1).

Based on information in the ARU Patch Readme and the Oracle Configurator About document, you may need to make adjustments to new functionality and changes that are introduced by the patch, such as disabling or changing the default value of new functionality.

If you are applying an Oracle Configurator patch that upgrades the Oracle Configurator release without upgrading the rest of Oracle Applications, you may encounter problems in reading and writing to the Oracle Applications database. If this occurs, the database administrator should check whether any indexes were dropped during the upgrade or any invalid objects exist in the database. These issues can affect performance when initializing Oracle Configurator and saving configurations. A database analysis can reveal such defects and areas of deadlock in the database transactions. Applying a subsequent patch can fix the problem. For more information, refer to Oracle Applications database administration documentation.

2.4.5.5 If Your User Community or Site Traffic Has Changed

It is important to plan for user volume appropriately and monitor site use. If your implementation recently became available to more clients or the number of users accessing your Web site has increased, the Web server configuration parameters may need to be modified. For details, see:

- Section 3.2, "Apache Web Server Parameters" on page 3-1
- Oracle Internet Application Server (*i*AS) documentation

2.4.5.6 If Your Configuration Model Uses Configurator Extensions

Configurator Extensions can be costly runtime events and as such may adversely affect performance. You may want to disable all Configurator Extensions temporarily before testing a configuration model to see whether they are the cause of or a contributing factor to poor performance. You can disable all Configurator Extensions by modifying the OC Servlet property `cz.activemodel`.

For example, edit `cz_init.txt` and add the following:

```
wrapper.bin.parameters=-Dcz.activemodel=/nofc|
```

If an entry for `cz.activemodel` already exists in `cz_init.txt`, simply append the "`/nofc|`" argument to the end to disable all Configurator Extensions. (This argument also disables all Functional Companions.)

For more information about `cz_init.txt` and `cz.activemodel`, see the *Oracle Configurator Installation Guide*.

2.5 Common Performance Tuning Questions

This section lists some questions that are commonly asked when tuning the performance of Oracle Configurator Developer or a runtime Oracle Configurator.

2.5.1 What is the recommended hardware setup?

See Section 2.6.2, "Hardware Recommendations" on page 2-15.

Other Ways To Phrase This Question

What are the minimum hardware requirements for Oracle Configurator?

2.5.2 What is the optimal way to set up the Web tier?

See Chapter 3, "Web Server Configuration" on page 3-1.

Other Ways To Phrase This Question

What is the recommended way to set up the Web server?

How does Oracle recommend setting up Apache and Oracle *i*AS?

2.5.3 How do I configure and tune the JServ to improve performance?

Oracle recommends that you dedicate at least one JServ to running only Oracle Configurator. For details, go to Metalink, Oracle's technical support Web site at:

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

Other Ways To Phrase This Question

Can I improve performance by modifying the Jserv?

2.5.4 Is there a recommended number of users per CPU?

See Section 2.6.2, "Hardware Recommendations" on page 2-15.

Other Ways To Phrase This Question

What is the recommended maximum number of users per CPU?

2.5.5 How can I optimize the performance of my Configurator Extensions?

The *Oracle Configurator Extensions and Interface Object Developer's Guide* contains suggestions on how to design Configurator Extensions for best runtime performance.

Other Ways To Phrase This Question

My Configurator Extensions take too long to execute. What can I do?

2.5.6 How can I improve initial configuration model load-time performance?

See Section 3.2.4, "Preloading the Oracle Configurator Servlet" on page 3-6.

Other Ways To Phrase This Question

Why does it take so long to load a configuration model?

2.5.7 What types of changes to my Model structure, rules, or UI will improve performance?

For details about tuning a configuration model, see the *Oracle Configurator Modeling Guide*.

Other Ways To Phrase This Question

What is the best way to design and build a configuration model for optimal runtime performance?

2.5.8 What configuration settings affect runtime performance?

Refer to the following sections for details:

- Section 2.6.3, "Profile Option Recommendations" on page 2-15
- Section 2.6.4, "Recommended Database Configuration Settings" on page 2-16
- Section 3.2, "Apache Web Server Parameters" on page 3-1

Other Ways To Phrase This Question

What are the recommended values for the various configuration settings?

2.5.9 How can I improve performance of Oracle Configurator Developer?

Perform the following to improve performance of Oracle Configurator Developer:

- Limit the amount of rows that appear in pages that display data in a hierarchical table by using the Focus icon and Folders effectively. For example, if you have many Models in the Main area of the Repository, consider grouping all related Models with into the same Folder.

  Use the Focus icon often to display only a specific object and its descendants (rather than all objects in the Repository or the entire Model structure, for example).

- Verify the value of the profile option CZ: Number of Rows Displayed in Hierarchical Tables is set to 50 (the default) or less. For more information about setting profile options, see the *Oracle Applications System Administrator's Guide*.
- Verify all profile options and Java system parameters that are related to logging are disabled. For more information, see "Troubleshooting" on page -xv.
- Click the Diagnostics global link, select Set Trace Level, and then click Go. Select Disable Trace, and then click Save.

Remember that many of the runtime performance issues described in this guide also apply to unit testing a configuration model from Configurator Developer (by clicking the Test Model button). Therefore, implementing suggestions in this guide, such as setting the maximum heap size appropriately and preloading Models, can also improve performance when unit testing.

2.5.10 How can I improve performance of the Generic Configurator User Interface?

The Generic Configurator UI displays only BOM Model items and enforces only implicit BOM rules. In other words, any Model structure nodes, rules, or UI elements that are defined in Configurator Developer are not available. Therefore, there is very little that can be done to improve performance when configuring a Model using the Generic Configurator UI. However, you may want to consider creating a UI for the Model in Configurator Developer, publishing the Model and UI, and then testing it to see if performance is better than the Generic Configurator UI.

Modifying the default values of profile options that are used by the Generic Configurator UI may improve performance when loading a configuration model.

The Generic Configurator UI uses the following profile options:

- CZ: BOM Tree Expansion State
- CZ: Generic Configurator UI Max Child Rows

For details, see Section 2.6.3, "Profile Option Recommendations" on page 2-15.

Other Ways To Phrase This Question

The Generic Configurator UI takes too long to appear. What can I do to improve performance?

2.5.11 How can I improve performance of the Oracle Order Management Batch Validation process?

Define the profile option CZ: Skip Validation Procedure. See the *Oracle Configurator Installation Guide* for details.

Consider modifying the `UtlHttpTransferTimeout` parameter in the CZ_DB_SETTINGS table. For details, see the *Oracle Configurator Implementation Guide*.

If you are not using Order Management to launch Oracle Configurator interactively, you may want to consider modifying the value of the profile option BOM: Configurator URL of UI Manager to point to a different JServ. By doing this, all batch validation requests made when placing an order will be directed to this alternative URL, reducing the amount of network activity on the JServ that is running Oracle Configurator for interactive users. See the *Oracle Configurator Installation Guide* for more information about this profile option.

Other Ways To Phrase This Question

Why is the Batch Validation process slow?

How can I segregate the Oracle Order Management Batch Validation process?

2.5.12 How can I improve performance when saving a configuration and returning to Oracle Order Management?

If the configuration model is large or has many BOM Model instances, setting the profile option CZ: Only Create CZ Config Items for Selected Nodes to Yes may improve performance. This profile option controls whether Oracle Configurator creates CZ Config Items (records) for options that are included in a saved configuration. This profile option is set to `No` by default. See the *Oracle Configurator Installation Guide* for more information.

This issue can also be addressed by improving performance of the host application. Refer to your Oracle Order Management documentation for more information.

2.5.13 How can I improve runtime performance when creating new component instances?

Create the desired number of instances before selecting other options on the page or configuring any instances.

For information about using Configurator Extensions to apply inputs, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

2.5.14 What affect does displaying prices and ATP data have on performance?

Displaying item prices and available to promise information can significantly affect runtime performance. Oracle recommends disabling prices and ATP data if they are not required.

For details, see Section 2.4.2, "Pricing and ATP" on page 2-8.

2.5.15 How can I improve runtime performance when selecting an option?

Performance issues that are observed when selecting a option are usually related to rules defined in Configurator Developer. The *Oracle Configurator Modeling Guide* contains information about designing a Model and rules for optimal runtime performance.

2.6 Performance Tuning Reference

This section provides information about the following areas:

- Software Recommendations
- Hardware Recommendations
- Profile Option Recommendations
- Recommended Database Configuration Settings

2.6.1 Software Recommendations

This section describes settings you can modify to improve performance of the application server and the Oracle Configurator Servlet, and lists the version of Java that Oracle Configurator supports.

Oracle *i*AS/Apache Configuration

For details about recommended Apache settings, see Section 3.2, "Apache Web Server Parameters" on page 3-1.

For details about setting Java heap size for optimal performance, see Section 3.2.2.1, "Heap Size" on page 3-3.

Oracle Configurator Servlet Parameters

Several Oracle Configurator Servlet parameters can affect performance. Recommended values for these parameters are provided in Section 3.2.2, "The cz_init.txt and jserv.properties Files" on page 3-2.

Java

Oracle recommends running version 1.5 of Java and the Java Development Kit (JDK).

Oracle recommends that you dedicate at least one JServ to running only Oracle Configurator. For details, go to Metalink, Oracle's technical support Web site at:

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

For details about setting Java heap size for optimal performance, see Section 3.2.2.1, "Heap Size" on page 3-3.

2.6.2 Hardware Recommendations

The primary factors that influence Oracle Configurator performance are listed in Section 1.2, "Factors that Influence Runtime Performance" on page 1-2. Because the number of concurrent Oracle Configurator users and the size and number of configuration models can vary widely from one installation to the next, it is impossible to recommend standard minimum hardware requirements.

However, Oracle recommends that you use a machine that provides the best cost to performance benefit. For example, Oracle Configurator performance is likely to improve considerably if you upgrade an older machine with a 1 GHtz CPU to a newer one with a 4 GHtz CPU.

2.6.3 Profile Option Recommendations

Table 2–1 lists the profile options that can affect runtime performance and settings that provide the best performance.

Refer to the *Oracle Configurator Installation Guide* for details about each profile option and to verify whether the setting recommended below is appropriate for your installation.

Table 2–1 Profile Options and Recommended Settings for Performance

Profile Option	Recommended Setting	Notes
CZ: BOM Tree Expansion State	One Level	This is the default value, and it should provide the best load-time performance. This profile option is used only by the Generic Configurator User Interface.
CZ: Generic Configurator UI Max Child Rows	50	This is the default value, but specifying a smaller value may improve load-time performance. This profile option is used only by the Generic Configurator User Interface.
CZ: Include Unchanged Install Base Items	No	This profile option is used only used when reconfiguring a installed instances. For more information, see the *Oracle Telecommunications Service Ordering Process Guide*.
CZ: Only Create CZ Config Items for Selected Nodes	Yes	Improves performance when saving large configuration models, or a configuration that has many initial BOM Model instances.
BOM: Configurator URL of UI Manager	The URL of a JServ that is running Oracle Configurator.	For details, see Section 2.5.11 on page 2-13.
CZ: Skip Validation Procedure	The name of a PL/SQL function that you define	See the *Oracle Configurator Installation Guide* for details.

2.6.4 Recommended Database Configuration Settings

To improve performance of the Oracle Order Management Batch Validation, modify the following settings in the CZ_DB_SETTINGS table:

- `AltBatchValidateURL`
- `UtlHttpTransferTimeout`

For more information about these settings, see the *Oracle Configurator Implementation Guide*.

3

Web Server Configuration

A well-tuned Web server is required for optimal performance from any Web-based application, including Oracle Configurator. There are several configuration parameters in the Apache Web server and the JServ module, including Configurator-specific parameters, which must be tuned to provide optimal performance.

Additionally, you can improve performance by load balancing the Apache Web listener (HTTPD) and adjusting heap size during startup.

For instructions in configuring, verifying, and—most importantly— troubleshooting the Apache Web server and the JServ module, see the *Oracle Configurator Installation Guide*.

See the "Structure" section in the Preface to learn about the audience for which this chapter is intended.

3.1 Overview

See Section 2.3, "Tuning Key Components of the Oracle Configurator Architecture" on page 2-4 for information about how the Web server participates in a deployed Configurator environment.

Tuning the Web server is an iterative process. Several parameters in various configuration files play a significant role in Oracle Configurator performance. Section 3.2, "Apache Web Server Parameters" presents these parameters, their role in runtime performance, and their default and recommended values. Tuning the Web server consists of modifying these values appropriately and iteratively.

In addition to setting parameter values in the configuration files, you can load balance the Apache Web listener (HTTP) or install multiple instances of the JServ engine to handle requests from multiple clients. For details about how to set up Apache for Oracle Configurator, see Oracle Internet Application Server release 1.0.2.2.2 documentation.

3.2 Apache Web Server Parameters

The high performance Apache Web server in the Oracle Internet Application Server (*i*AS) is part of the Oracle Configurator architecture. Apache Web server parameters are set in the following configuration files:

- The jserv.conf File
- The httpd.conf and httpds.conf Files

For a summary of the Apache Web server parameters that affect Oracle Configurator performance, their recommended values, and the file in which each parameter is set, see Appendix A, "Summary of Web Configuration Parameters".

For details about correct configuration and location of these Web server files for running Oracle Configurator, see the *Oracle Configurator Installation Guide*.

For additional information about Oracle *i*AS, see Oracle Internet Application Server (*i*AS) documentation.

3.2.1 The jserv.conf File

The `jserv.conf` file contains parameters that define your Apache and JServ configuration. This file is set up automatically when you run RapidInstall.

Check the following parameters and modify their values as necessary.

3.2.1.1 ApJServManual

`ApJServManual` contributes to performance by controlling whether the Apache Web listener can be load balanced.

Default Value

The ApJServManual parameter is set to `Off` by default. The `Off` value ignores load balancing. When you are using load balancing, the value of this parameter must be `On`.

Recommended Value

The recommended value is `On`.

3.2.1.2 ApJServVMTimeout

`ApJServVMTimeout` contributes to performance by affecting the amount of time to wait for the JVM to start up or respond. If this value is too low, the JVM shuts down when left idle. Slow or heavily loaded machines benefit from a high value.

Default Value

The `ApJServVMTimeout` parameter is set to `10` seconds by default, which is also the timeout if this parameter is commented out in the `jserv.conf` file.

Recommended Value

The recommended value is `1800` seconds (30 minutes).

3.2.2 The cz_init.txt and jserv.properties Files

The only Oracle Configurator-specific property in `jserv.properties` that is read by the OC Servlet is `cz_properties_file`. This property identifies the location of the file `cz_init.txt`. Any servlet properties that you have customized (for example, by modifying their default values) must be defined in `cz_init.txt`.

Other properties defined in `jserv.properties` are provided only to support legacy User Interfaces (DHTML and Java applet) that were created in previous versions of Oracle Configurator Developer.

The following OC Servlet properties are specific to Oracle Configurator and affect runtime performance:

- cz.uiservlet.dio_share on page 3-4

- cz.uiservlet.pre_load_filename on page 3-4
- cz.activemodel on page 3-5
- cz.uiserver.poll_timeout_applet on page 3-5
- cz.uiserver.check_heartbeat_timeout on page 3-5

For information about other OC Servlet properties and how to set them, see the *Oracle Configurator Installation Guide*.

3.2.2.1 Heap Size

The C and Java code of Oracle Configurator run in the Web server. Therefore, it is important to consider the heap size allocation during startup.

The initial Java parameters for Linux™ (RHEL 3 or higher) and JDK 1.4.2 or higher should be:

```
-Xms600m; initial heap
-Xmx2000m; max heap
-Xss1m; native stack size
```

These options are non-standard, subject to change, and vary from platform to platform. Specify a small initial heap and a maximum heap that is smaller than the platform maximum. This is necessary to minimize the chances of running out of memory for the C heap.

Use one of the following methods to determine the maximum heap size for your platform:

- On Windows, set the maximum heap to 1300m and the minimum heap to 500m
- Consult your platform vendor documentation or Web page for information regarding the maximum heap size for the JVM version you are using

You can determine the maximum heap using a binary search for your platform and JDK combination by running Java from the command line and specifying the maximum heap. For example:

```
java -mx3000m -version
```

You may see a message similar to the following:

```
Error occurred during initialization of VM
Could not reserve enough space for object heap
```

In this case, decrease the maximum heap value until the current JDK version appears, but the error message does not. For example:

1. Start at 1000M.
2. If that works, enter a higher value, such as 3000M.
3. If 3000M doesn't work, set it to 2000M.
4. Continue until you find your platform/JDK maximum heap size.

Once you have determined the maximum heap size, specify it in your `Jserv.properties` file such that the Java maximum heap is either the platform maximum or 3GB, whichever is smaller. For example, your `Jserv.properties` entry might be:

```
wrapper.bin.parameters=-Xms500M -Xmx3000 -XX:NewSize=128M -XX:MaxNewSize=128M
```

3.2.2.2 cz.uiservlet.dio_share

See the *Oracle Configurator Installation Guide* for basic information about this parameter.

The `cz.uiservlet.dio_share` parameter contributes to performance by improving the load performance of configuration sessions after the initial load for a given configuration model. The `cz.uiservlet.dio_share` parameter should be set to true when benchmarking.

The setting of this parameter interacts with the setting of `czservlet.pre_load_filename` and `cz.uiserver.lazyload`. For instance, `czservlet.pre_load_filename` essentially precaches the model, making the initial load no longer than subsequent load times. On the other hand, setting `cz.uiserver.lazyload` may minimize the advantages of preloading the model.

Setting this parameter to `false` disables sharing the cached configuration model, which implies that every user experiences the initial load time.

Default Value

The `cz.uiservlet.dio_share` parameter is set to `true` by default.

Recommended Value

For better runtime performance, the recommended value is `true`. Even when this parameter is set to `true`, testing configuration models from Oracle Configurator Developer always reloads the configuration models so that changes are visible.

3.2.2.3 cz.uiservlet.pre_load_filename

Defining the OC Servlet parameter `czuiservlet.pre_load_filename` can improve performance by caching one or more configuration models when the OC Servlet starts up.

If loading Models for specific languages, then `database_id` and the `icx_session_ticket` parameter are required for the database session in that language. For details about these parameters, see the *Oracle Configurator Implementation Guide*.

> **Note:** If you are preloading a configuration model that uses an HTML UI (a UI based on the Oracle Applications Framework), the OC Servlet takes longer to start up the first time a user starts a configuration session. However, the initial response time should improve noticeably for subsequent users.

Default Value

The `czuiservlet.pre_load_filename` parameter is not set by default.

Recommended Value

For best performance, the recommended value of this parameter is the absolute path to a file containing the initialization message that preloads a configuration model(s) so the OC Servlet is initialized when the first user starts a configuration session. Note that the parameter `cz.uiservlet.dio_share` must be set to `true` to cache configuration models.

See the *Oracle Configurator Installation Guide* for detailed information about how to use `czuiservlet.pre_load_filename` and its interaction with other parameters such as `cz.uiservlet.dio_share` and `servlets.startup`.

Additionally, see MetaLink Note 338841.1 for information about `czuiservlet.pre_load_filename` that is not included in the *Oracle Configurator Installation Guide*.

3.2.2.4 cz.activemodel

The `cz.activemodel` parameter enables you to suppress display of prices, which can significantly affect performance. This parameter can also be used to disable all Configurator Extensions and Functional Companions.

The switches for displaying list prices, discount prices, and ATP data should only be turned on if it is essential to display prices in a runtime Oracle Configurator.

See the *Oracle Configurator Installation Guide* for basic information about `cz.activemodel`, including examples of how the settings for its various switches affect pricing and ATP at runtime.

See the *Oracle Configurator Implementation Guide* for more information about displaying pricing and ATP in Oracle Configurator.

Default Value

By default, the `cz.activemodel` parameter is set to `/nolp|/nodp|/noatp|`. This means that list prices, discount prices, and ATP data are *not* displayed.

Recommended Value

For best performance, accept the default values for this parameter (in other words, do not display prices and ATP data).

3.2.2.5 cz.uiserver.poll_timeout_applet

See the *Oracle Configurator Installation Guide* for basic information about the "heartbeat" mechanism and how this parameter interacts with the parameter `cz.uiserver.check_heartbeat_timeout`.

Default Value

The `cz.uiserver.poll_timeout_applet` parameter is set to `10000` milliseconds by default.

Recommended Value

The recommended value is 10000 milliseconds.

3.2.2.6 cz.uiserver.check_heartbeat_timeout

See the *Oracle Configurator Installation Guide* for basic information about the "heartbeat" mechanism and how this parameter interacts with the parameters `cz.uiserver.poll_timeout_applet` and `cz.uiserver.heartbeat_interval`.

Default Value

The `cz.uiserver.check_heartbeat_timeout` parameter is set to `30000` milliseconds by default.

Recommended Value

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

3.2.2.7 cz.runtime.use_dedicated_jvm

This parameter controls whether Oracle Configurator runs on the same server as the other Oracle Applications products you have installed.

Default Value

By default, the `cz.runtime.use_dedicated_jvm` parameter is not defined and Oracle Configurator runs on the same server as other Oracle Applications products.

Recommended Value

Oracle strongly recommends adding this parameter to the `cz_init.txt` file and setting it to `true`. Then, set up a JServ that is dedicated to running only Oracle Configurator. For details, go to Metalink, Oracle's technical support Web site at:

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

For more information about defining OC Servlet parameters, see the *Oracle Configurator Installation Guide*.

3.2.3 The httpd.conf and httpds.conf Files

See the *Oracle Configurator Installation Guide* for basic information about these files and how they are used to configure the Apache Web listener in *i*AS. Both files contain the same information, including the directives: `Port`, `Documented`, `Alias`, and `Includes`.

The difference between these two files is that `httpds.conf` is used when there is a secure socket.

The Timeout parameter is located in the `httpd.conf` or `httpds.conf` file, and can affect performance.

3.2.3.1 Timeout

The `Timeout` parameter contributes to performance by allowing you to control how long Apache waits for receipt of TCP packets on a POST request, or how long it takes to receive a GET request. This means that when loading a configuration model, there is no response back to the client until the `Timeout` parameter's value is reached, or the JServ responds, whichever comes first. With large models loading into the JServ, you will need a correspondingly long timeout to keep the HTTP listener waiting for a response until the configuration model is loaded.

Default Value

The `Timeout` parameter is set to `300` seconds (5 minutes) by default.

Recommended Value

The recommended value is `3600` seconds (60 minutes). The value should be at least `1800` seconds (30 minutes) for adequate Oracle Configurator performance. There is no performance loss from having the `Timeout` parameter set to a large value, unless the JServ stops responding.

3.2.4 Preloading the Oracle Configurator Servlet

You can improve performance when initializing the Oracle Configurator Servlet by preloading one or more configuration models. Initializing refers to loading a configuration model for the first time during an OC Servlet session. Preloading a

configuration model means loading data into cache memory before launching Oracle Configurator to improve performance.

For details about preloading a configuration model, refer to Section 3.2.2.3, "cz.uiservlet.pre_load_filename" on page 3-4.

3.2.5 Load Testing

To prepare your application for realistic conditions, you should use a testing tool that allows you to set up reasonable tests that emulate what your end users will do.

There are various types of testing you should employ, such as:

- Stress Testing
- User Activity Testing

3.2.5.1 Stress Testing

In stress testing, the testing tool fires as many commands as possible at the server to test CPU utilization rate. Try to measure the number of hits per second you can trigger while keeping the CPU utilization under 85% at peak loads. If you are testing load and response behavior, you may want to test up to 100% CPU utilization, then observe the behavior and plot system scalability.

3.2.5.2 User Activity Testing

In user activity testing, you use a tool to build test scripts that represent what your end users will be doing with the kinds of delays that a user would experience.

After creating scripts that emulate users, run that number of users against the server, at randomly sequenced times, to represent a close approximation of what the application will handle.

Oracle strongly recommends using a scripting tool such as LoadRunner to perform user activity testing. If you are not using a scripting tool, the only way to extract load times, event timings, and so on is by reviewing log files, which can be a difficult and time consuming task.

Examples

- The user might pause and click Next, Back, and Next several times.
- The user might click a number of checkboxes rapidly, because he knows what he wants, since he has previously visited that part of your application.
- The user might pause for a period of time between each click.

The variability of these different scripts, all running within the same time span, gives you a good sense of what your system can actually handle in terms of number of users.

3.2.6 Load Balancing

Oracle strongly recommends using Oracle Internet Application Server (iAS) version 1.0.2.2.2 or later. Version 1.0.2.2.2 can be set up to use a process manager that automatically load balances server processes. This version also supports Secure Sockets Layer (SSL).

For more information, refer to Oracle Internet Application Server release 1.0.2.2.2 documentation.

4

Tools and Collecting Data

The goal of collecting data is to determine whether the production architecture supports all possible end users accessing every available configuration model.

Use a benchmarking tool to test the performance of your configuration model(s) and determine software and hardware sizing that is appropriate for the full range of possible end user scenarios.

> **Note:** Benchmarking tools cannot account for the client-side rendering time, which may be very important.

The sections in this chapter are:

See the "Structure" section in the Preface to learn about the audience for which this chapter is intended.

4.1 Data Collection and Analysis

There is a wide range of data that you can collect for analysis to identify modifications that could improve performance.

4.1.1 Areas To Examine

Areas that may yield useful data for performance analysis include:

- Database or database server size and setup
- Client machine size and setup
- Server machine size and setup
- Host application session time
- Configuration session data

Another area that may yield important data for analysis is end-user response time. End-user response time is the sum of the server response time, the display time, network latency, and think time. This is the total throughput under load. There is an amount of time that end users will wait for an application to respond before moving on to something else. That end-user walk away time depends on the business and the task. The end-user response time must be less than the end-user walk away time.

For a more specific listing of areas to examine, see Section 2.3, "Tuning Key Components of the Oracle Configurator Architecture" on page 2-4 and Section 2.4, "Isolating Performance Problems and Intervening" on page 2-7.

4.1.2 Benchmarking

Benchmarks allow you to compare performance measurements across changes in the configuration model, machine and server configuration, and other elements affecting performance.

Benchmarks also allow you to size your hardware appropriately to best match your current requirements, as well as handle your future requirements.

Planning and executing the appropriate benchmark tests of any distributed computing system is both critical and complex. Fundamentally, any benchmark you undertake should reflect the actual deployment environment. This means that it should be carried out:

- With the appropriate server-class hardware platform
- Over a long duration
- With multiple end users

For example, if your deployment involves dial-up users, your benchmark should test for the slowest expected dial-up speeds.

WARNING: Do not benchmark using the Test button in Oracle Configurator Developer. The Test button invokes initialization code that sets the internal equivalent of the `cz.uiservlet.dio_share` **parameter to false, which causes every end user to experience the initial load time. Using an HTML launch page or the host application to test Oracle Configurator requires explicitly setting the** `cz.uiservlet.dio_share` **parameter.**

Conducting a benchmark involves answering the following:

- What is important to measure?
- How are the results to be interpreted?
- What is the minimum necessary test duration?
- How often should the measurements be sampled?
- How can a realistic deployment be constructed for the purposes of measuring a benchmark?

4.1.3 Analyzing Data

You analyze data to determine performance and where to tune your system. The data comes from your careful documentation of benchmark test conditions and results, including the data collected by an analysis tool such as LoadRunner. Data analysis often involves making calculations that provide the limits within which your system can perform acceptably, such as the number of concurrent sessions a single CPU can support.

When analyzing your data, keep in mind that approximately 30% CPU utilization needs to be available for running operating system processes. That is equivalent to tuning the runtime Oracle Configurator performance to 70% CPU utilization. Tests

with different numbers of end users can help you determine how many end users 70% CPU utilization supports. For details about how components of the Oracle Configurator architecture compete for CPU utilization, see Section 2.3.2, "Server Machine Tuning" on page 2-6 and Section 2.4.1, "System Transactions" on page 2-7.

You can focus on the which runtime events require large numbers of transactions (and therefore processing) by comparing the number of transactions logged by your analysis tool.

When determining total client rendering time from your raw data, it is necessary to add the elapsed model update time to the elapsed layout time. See Section C.2.1, "Client Side Data" on page C-2 for details about collecting this data.

4.2 Data Collection Timing and Scope

Be sure to conduct tests that include the hosting application, such as *i*Store.

Collect performance numbers for individual operations within a configuration session, such as screen flips and option clicks.

Collect data when the most useful performance data occurs, which is when 90% of the end users are actively engaged in a configuration session. For this reason, the case studies documentation shows the response time on various operations for the 90th percentile of the full load.

When an application goes into production, it will likely be operating on a nonstop (24X7) schedule. Any benchmark you undertake should be designed to reflect these requirements, and as such should be run over a period of time. A useful benchmark should run at least 24 hours, longer if possible.

4.2.1 Server Side Data

While collecting server side performance numbers, keep several recommendations in mind:

1. CPU utilization should not exceed 85%.
2. If the end-user response time is unacceptable when CPU utilization is below 85%, decrease concurrent users or repeat the tuning process to obtain better response times.

For additional details about collecting server side data, see Section 4.3, "Tool: LoadRunner" on page 4-3.

4.3 Tool: LoadRunner

LoadRunner is one of the many tools used to perform load testing. Load testing involves stressing the product or observing server side performance under specific loads. You can use LoadRunner to collect performance numbers for individual operations within a configuration session.

Use LoadRunner's Analysis tool to get such data as:

- Response time
- CPU utilization
- Number of transactions per hour
- Throughput
- Hits per second

Figure 4–1 Screen Capture of LoadRunner CPU Utilization Analysis

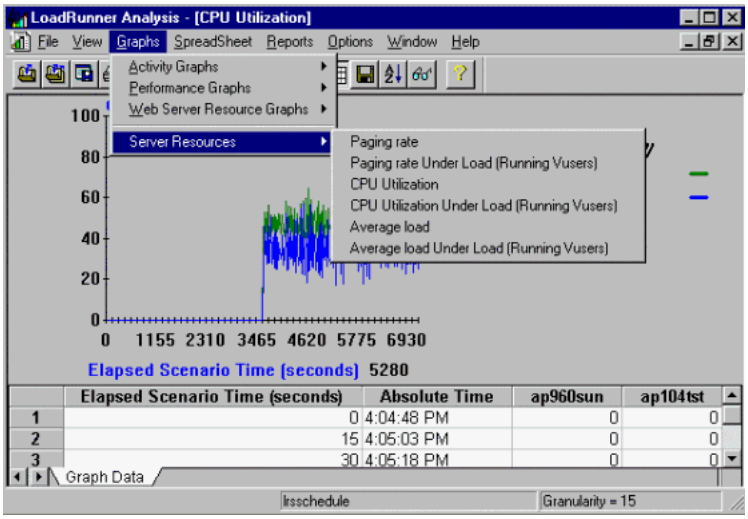
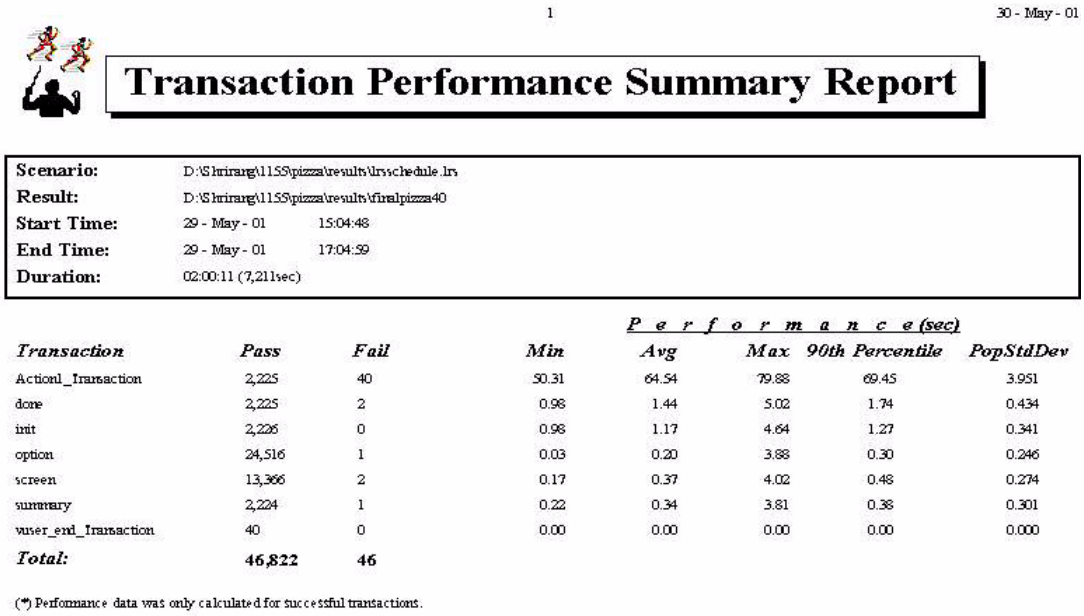


Figure 4–2 Example of a LoadRunner Transaction Performance Summary Report



1 30 - May - 01

Transaction Performance Summary Report

Scenario:	D:\Shrirang\11SS\pizza\results\lrsschedule.lrs
Result:	D:\Shrirang\11SS\pizza\results\finalpizza40
Start Time:	29 - May - 01 15:04:48
End Time:	29 - May - 01 17:04:59
Duration:	02:00:11 (7,211sec)

			Performance (sec)				
Transaction	*Pass*	*Fail*	*Min*	*Avg*	*Max*	*90th Percentile*	*PopStdDev*
Action1_Transaction	2,225	40	50.31	64.54	79.88	69.45	3.951
done	2,225	2	0.98	1.44	5.02	1.74	0.434
init	2,226	0	0.98	1.17	4.64	1.27	0.341
option	24,516	1	0.03	0.20	3.88	0.30	0.246
screen	13,366	2	0.17	0.37	4.02	0.48	0.274
summary	2,224	1	0.22	0.34	3.81	0.38	0.301
vuser_end_Transaction	40	0	0.00	0.00	0.00	0.00	0.000
Total:	**46,822**	**46**					

(*) Performance data was only calculated for successful transactions.

4.3.1 Settings Specific to Oracle Configurator

For details about setting LoadRunner parameters that are specific to Oracle Configurator and necessary for collecting Oracle Configurator performance data, see Appendix B, "LoadRunner Settings".

4.3.2 General Tips and Guidelines

Load simulation does not require running a wide variety of test cases. Select a group of test cases that represent different usages of the application (such as configuring small, medium, and large systems). There is no need to have multiple test cases representing different input values for one particular use of the application.

Use only test cases that succeed in saving a configuration because each user should go through multiple iterations of the test scripts.

Have written test plans to follow because several recording iterations may be required and repeatability is critical to the correctness of the scripts. During LoadRunner recording sessions, you see the captured commands inserted in the Vuser scripts in real time as you walk through the test plan.

To execute the Vuser scripts for Oracle Configurator testing, see Appendix B, "LoadRunner Settings".

A

Summary of Web Configuration Parameters

Table A–1 lists the parameters that are described in detail in Chapter 3, "Web Server Configuration" in alphabetical order, indicating the recommended value, what file contains the parameter, and where to turn for additional information.

Table A–1 Web Server Parameters That Affect Oracle Configurator Performance

Parameter	Recommended Value	File	Details
ApJServManual	`On`	`jserv.conf`	Section 3.2.1.1 on page 3-2
ApJServVMTimeout	`1800`	`jserv.conf`	Section 3.2.1.2 on page 3-2
cz.activemodel	`/nolp|/nodp|/noatp|`	cz_init.txt	Section 3.2.2.4 on page 3-5
cz.uiservlet.dio_share	`true`	cz_init.txt	Section 3.2.2.2 on page 3-4
cz.uiserver.check_heartbeat_timeout	# of milliseconds it takes to load the model	cz_init.txt	Section 3.2.2.6 on page 3-5
cz.uiserver.poll_timeout_applet	`20000` to `60000`	cz_init.txt	Section 3.2.2.5 on page 3-5
cz.uiservlet.pre_load_filename	text file with initialization message	cz_init.txt	Section 3.2.2.3 on page 3-4
cz.runtime.use_dedicated_jvm	true	cz_init.txt	Section 3.2.2.7 on page 3-6
Timeout	`3600`	httpd.conf httpds.conf	Section 3.2.3.1 on page 3-6

B

LoadRunner Settings

This appendix describes some LoadRunner parameters that are specific to Oracle Configurator and are necessary for collecting Oracle Configurator performance data. The sections are:

- Virtual User (Vuser) Generator
- Vuser Parameterization
- Modifying Vuser Scripts
- Debugging Vuser Scripts

This information is based on LoadRunner 8.0. LoadRunner consumes significant system resources, so it must be installed and run on a separate machine from the Web server being tested for load performance.

For information about LoadRunner settings generally, see the LoadRunner documentation listed in the Section "Related Documents" in the preface of this guide.

For additional tips and guidelines on running tests, see Section 4.3.2, "General Tips and Guidelines".

See the Section "Structure" on page xii in the preface to verify the audience for whom this chapter is intended.

B.1 Virtual User (Vuser) Generator

Before recording the Vuser scripts, log in to LoadRunner selecting the Vuser type "Oracle Web Applications 11*i*", and set parameters described in this section. Selecting this Vuser type loads the appropriate libraries for recording Vuser scripts emulating Oracle Configurator users.

- Tools Menu: Recording Options
- Tools Menu: General Options
- Vuser: Run-Time Settings

B.1.1 Tools Menu: Recording Options

From the menu toolbar, select Tools: Recording Options. Make the following section in the tabs of the Recording Options window.

Browser Section

Specify the path to Netscape or Internet Explorer browser.

Recording Proxy Section

Select **Obtain the proxy settings from the recording browser**. Make sure your browser proxy is set up correctly, especially if you need to traverse a firewall.

Recording Section

Select **URL-based Script**. Click the **URL Advanced** button, and then select the following options:

- **Reset Context for each action**
- **Create concurrent groups for resources after their source HTML Page**
- **Use web_custom_request only**

Correlation Section

Select **Enable Correlation during recording**. After the script is recorded, run the script twice. This will parameterize the transaction ID and the session ID. Then, click Find Correlations.

Advanced Section

Select **Reset context for each action**.

B.1.2 Tools Menu: General Options

From the Menu toolbar, select Tools: General Options. Make the following selection in the tabs of the General Options window.

Display Tab

Show browser during replay should not be selected. The runtime viewer does not support JavaScript, so it does not correctly display the screens for Oracle Configurator during replay.

B.1.3 Vuser: Run-Time Settings

Run-Time Settings are used to specify the runtime settings for the Vuser scripts. These settings are associated with Vuser scripts. You can modify the runtime settings in this section or from the LoadRunner Controller tool.

From the Menu toolbar, select Vuser: Runtime Settings. Make the following selections in the Run-Time Settings window.

Run Logic Section

When debugging or during single user scenarios, initially set **Number of Iterations** to `1`. When simulating multi-user loads, set **Number of Iterations** to a large number such as `100`. In the **Pacing** section, set **Iteration Pace** to **As soon as the previous iteration ends**. If iterations can't be started at a specified time, it may not be effective or useful to select **Start new iteration**.

Log Section

Select **Standard log** initially or during debugging. Disabling logging allows the script to run faster. **Extended log** is not necessary unless you suspect a bug in LoadRunner or need more information for debugging.

Think Time Section

Set **Use a percentage of recorded think time** to the range of 50% to 200% for random think time. During debugging, it is appropriate to select **Ignore think time**.

Network Section

In the **Speed Simulation** section, select **Use Maximum Bandwidth**.

Browser Emulation Section

Select the appropriate browser software version from the User Browser list of values. If the list of values does not contain your browser software version, select **Use Custom Browser** and then enter the appropriate identification. For example, for Mozilla 4.73, enter:

Mozilla/4.73 [en] (WinNT; I)

`Http 1.1` is not compatible with the **Advanced Options** used. The custom user agent information is passed to the Apache server so that the right template is returned to the client (such as Internet Explorer or Netscape). This data can be found in the Oracle Configurator session log file during an interactive session (search for `user-agent` in the log).

Proxy Section

Select **Obtain the proxy settings from the default browser**.

Miscellaneous Section

In the **Multithreading** section, select **Run Vuser as a thread**. No other options should be selected.

B.2 Vuser Parameterization

To make the Vuser scripts more portable for tests running on different machines, parameterize the host and port information. See the LoadRunner documentation listed in the Section "Related Documents" in the preface of this guide for details.

B.3 Modifying Vuser Scripts

The following modifications of recorded Vuser scripts are specifically useful to LoadRunner testing of Oracle Configurator.

B.3.1 Think Times

LoadRunner records all pauses and many `lr_think_time()` commands are inserted into a Vuser script during recording. You only need one `lr_think_time()` command at the beginning of the script and between each transaction, so clean out all the excess think time commands. This better simulates a real user pausing between mouse clicks.

B.3.2 Optimize Vuser Scripts

The start of each test script's `Action()` section should contain the following commands:

```
web_cleanup_cookies();
web_set_max_retries("0");
web_set_timeout (RECEIVE, "300");
```

The `web_cleanup_cookies()` command makes sure the previous sessionID is cleared before starting a new configuration session (before the `Init` transaction).

The `web_set_max_retries("0")` command prevents the Vuser script from trying to connect to the server again if the first connection request failed.

The `web_set_timeout()` command sets the timeout limit (also a runtime setting).

Example B–1 shows how to segment scripts into transactions to get the performance response times for specific actions, such as selecting an Option.

Example B–1 Transactions in Test Scripts

```
lr_think_time( 3 );

lr_start_transaction("OptionClick");

web_add_header("Content-Type", "application/x-www-form-urlencoded");
web_custom_request("oracle.apps.cz.servlet.UiServlet",
"URL=http://{Host}:{Port}/servlets/oracle.apps.cz.servlet.UiServlet",
"Method=POST",
"TargetFrame=",
"Resource=0",
"RecContentType=text/html",
"SupportFrames=0",
"Body=XMLmsg=%3Cclient-event+session-id%3D%27{Session_
Id}%27%3E%3Cinput+rtid%3D%27294%27%3E1%3C%2Finput%3E%3C%2Fclient-event%3E&sessionI
d={Session_Id}",
LAST);

lr_end_transaction("OptionClick", LR_AUTO);
```

B.4 Debugging Vuser Scripts

The only way to check correctness of a script's execution is to put validation checks into the test script itself. For a benchmark, have all configuration sessions end in a saved and valid configuration. Place checks at the end of the script for complete and saved configuration status.

1. Declare the following variables at the beginning of the `Action()` section:

```
LPCSTR vuser_group;
int Vuser;
```

2. Before the `Save` transaction, insert the following:

```
/*
* Determine test success through these variables:
* config_hdr_id, is_valid, is_complete, and exit_status.
*/

web_create_html_param("config_hdr_id",
"id=\"_czt1001-HeaderIdValue\" class=\"x2\">",
"</span>");
web_create_html_param("is_valid",
"id=\"_czt1001-ConfigurationValidValue\" class=\"x2\">",
"</span>");
web_create_html_param("is_complete",
"id=\"_czt1001-ConfigurationCompleteValue\" class=\"x2\">",
```

```
"</span>");
web_create_html_param("exit_status",
"id=\"_czt1001-ConfigurationExitStatusValue\" class=\"x2\">",  "</span>");
```

3. After the `Save` transaction, insert the following:

```
/*
*  Test completed successfully only when is_valid is true, is_complete is true,
*  and exit_status is save.  Otherwise, the test failed.
*  No other verifications are done in the script to slowdown execution.
  */
 /* Get vuser number and vuser group name. */
 lr_whoami( &vuser,&vuser_group,NULL );
 if ( (strcmp(lr_eval_string("{is_valid}"), lr_eval_string("true")) ==
0) &&
   (strcmp(lr_eval_string("{is_complete}"), lr_eval_string("true")) ==
0) &&
(strcmp(lr_eval_string("{exit_status}"), lr_eval_string("save")) ==
0) )
{
lr_log_message("+++++ Config Hdr ID %s done by VUser %d in group
%s.", lr_eval_string("{config_hdr_id}"), vuser, vuser_group);
}
```

Even if the test succeeds, a validation check error message is written to the LoadRunner Controller window and log file. If there is a way to directly assert a test failure, place it inside the else statement after the `Save` transaction in the Vuser script.

C

DHTML User Interfaces

Oracle Configurator supports DHTML User Interfaces that are created in a previous version of Oracle Configurator Developer. This appendix provides suggestions that apply only to improving runtime performance of DHTML UIs.

Information about maintaining DHTML UIs is available in the *Oracle Configurator Installation Guide*.

C.1 Network Tuning

Network latency between the client and server could be an issue if there is a significant difference between client-side response times (recorded in the `czSource.jsp` file) and the time the same action takes on the Web server (recorded in the appropriate `cz_session` log file). See Section C.2.1, "Client Side Data" on page C-2 for details.

For additional performance improvement in the area of rendering UI controls, screens, and propagations of selection changes in the Oracle Configurator window, attention to network tuning between the client and the Web server could be helpful.

Although network tuning information is not within the scope of this book, see Section 2.3.1.6, "The Network" on page 2-6 for suggestions.

C.2 Client Machine Tuning

The client machine configuration is not generally under the control of the Web-based Oracle Configurator in a typical business-to-consumer production environment. However, it is possible to include guidance on the host application Web site recommending machine and browser specifications that provide the best performance.

When client machines *are* under your control, such as during testing or for internal deployments, tune the client machines for optimal performance. Tuning the client machine configuration is likely to produce performance improvements.

The sequence of tuning tasks within the client machine setup should be, in order:

- Browser settings
- Swap space
- Memory

If you have control over the client machine setup, you can improve client-side performance by:

- Setting up adequate swap space and keeping disks de-fragmented
- Increasing memory, virtual memory, or upgrading the CPU

- Verify that browser settings support Javascript (see the *Oracle Configurator Implementation Guide* for details)
- Tuning browser settings, browser types, browser versions

If performance has degraded, compare the performance on a changed or different client machine to a machine where performance was acceptable. If additional software was installed on the client machine, this could affect performance. If other applications are also running more slowly, this could be an indication of a full disk or insufficient memory.

C.2.1 Client Side Data

The information presented in this section is related to the tasks described in Chapter 4, "Tools and Collecting Data", but is specific to DHTML UIs. Therefore, it is recommended that you review Chapter 4 before reading this section.

You can collect client-side rendering time by setting the debug mode to true in the `czSource.jsp` file (set `bDebugMode = true`).

Note: Since setting `bDebugMode - true` involves modifying the `czSource.jsp` file on the middle tier where Oracle Configurator is installed, it is best to collect this client-side data in a test environment.

When `bDebugMode - true`, a window opens on the client machine (see Figure C–1). For each action taken in the Oracle Configurator window, the debug mode displays the following in the `czSource.jsp` file.

Elapsed Time Laying Out Controls and Screens

Elapsed Layout Time in `the czSource.jsp` file is the time taken to paint the UI controls and screens.

Elapsed Time Updating Model

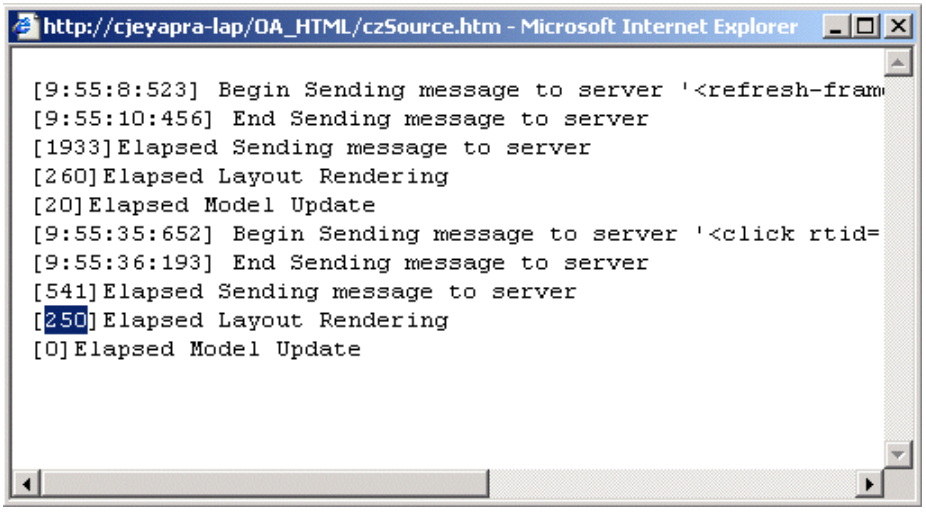
Elapsed Model Update Time in `the czSource.jsp` file is the time taken to update the UI controls with values.

Elapsed Time Sending Message to Server

Elapsed sending message to server time in `the czSource.jsp` file is the time taken to send a message from the client browser to the Web server (network time).

The Total Client Rendering Time is the sum of Elapsed Layout Time and the Elapsed Model Update Time.

Figure C–1 Elapsed Times in `czSource.jsp` File



You will have to record which rtid in the `czSource.jsp` file maps to which end user action in the user interface. The **Begin Sending message to server** times, both at session initialization and at the submit or done action, are partial times, so are not as directly meaningful as the elapsed times on option clicks and screen flips.

During a LoadRunner benchmark test, set the debug mode to false in the `czSource.jsp` file (set `bDebugMode = false`). LoadRunner can only simulate end users and does not start an interactive client session, so the client side data cannot be captured during a benchmark test.

Glossary

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

API

Application Programming Interface

applet

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

Archive Path

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

argument

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

ATO

Assemble to Order

ATP

Available to Promise

base node

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

bill of material

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

Bills of Material

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

binding

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

BOM

See **bill of material**.

BOM item

The **node** imported into **Oracle Configurator Developer** that corresponds to an Oracle **Bills of Material** item. Can be a **BOM Model**, **BOM Option Class node**, or **BOM Standard Item node**.

BOM Model

A model that you import from Oracle **Bills of Material** into **Oracle Configurator Developer**. When you import a BOM Model, effective dates, **ATO** rules, and other data are also imported into Configurator Developer. In Configurator Developer, you can extend the structure of the BOM Model, but you cannot modify the BOM Model itself or any of its attributes.

BOM Model node

The imported **node** in **Oracle Configurator Developer** that corresponds to a **BOM Model** created in Oracle **Bills of Material**.

BOM Option Class node

The imported **node** in **Oracle Configurator Developer** that corresponds to a BOM Option Class created in Oracle **Bills of Material**.

BOM Standard Item node

The imported **node** in **Oracle Configurator Developer** that corresponds to a BOM Standard Item created in Oracle **Bills of Material**.

Boolean Feature

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

bug

See **defect**.

build

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

CDL

See **Constraint Definition Language**.

CIO

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

command event

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

Comparison Rule

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

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

Compatibility Rule

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

Compatibility Table

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

component

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

Component

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

Component Set

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

concurrent program

Executable code (usually written in SQL*Plus or Pro*C) that performs the function(s) of a requested task. Concurrent programs are stored procedures that perform actions such as generating reports and copying data to and from a database.

configuration

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

configuration attribute

A characteristic of an **item** that is defined in the **host application** (outside of its inventory of items), in the **Model**, or captured during a **configuration session**. Configuration attributes are inputs from or outputs to the host application at initialization and termination of the configuration session, respectively.

configuration engine

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

Configuration Interface Object

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

configuration model

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

configuration rule

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

configuration session

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

configurator

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

Configurator Extension

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

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

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

Configurator Extension Archive

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

connectivity

The connection between client and database that allows data communication.

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

Connector

The **node** in the **model structure** that enables an **end user** at **runtime** to connect the Connector node's parent to a referenced **Model**.

Constraint Definition Language

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

Container Model

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

Contributes to

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

Consumes from

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

count

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

CTO

Configure to Order

customer

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

customer requirements

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

CZ

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

CZ schema

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

data import

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

data source

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

DBMS

Database Management System

default

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

Defaults relation

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

defect

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

Design Chart

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

developer

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

Developer

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

DHTML

Dynamic Hypertext Markup Language

discontinued item

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

element

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

end user

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

enterprise

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

environment

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

ERP

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

event

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

Excludes relation

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

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

feature

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

Feature

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

functional specification

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

generated logic

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

guided buying or selling

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

host application

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

HTML

Hypertext Markup Language

implementation

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

implementer

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

Implies relation

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

import server

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

import tables

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

initialization message

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

Instance

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

instance

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

Also, the memory and processes of a database.

instantiate

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

integration

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

integration testing

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

item

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

Item

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

Item Master

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

Item Type

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

Java

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

Java class

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

JavaServer Pages

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

JSP

See **JavaServer Pages**.

legacy data

Data that cannot be imported without creating custom extraction programs.

listener

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

load

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

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

log file

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

Logic Rule

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

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

maintainability

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

maintenance

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

Metalink

Oracle's technical support Web site at:

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

method

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

Model

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

model

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

model-driven UI

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

model structure

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

Negates relation

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

node

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

Numeric Rule

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

object

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

OC

See **Oracle Configurator**.

OCD

See **Oracle Configurator Developer**.

option

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

Option

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

Oracle Configuration Interface Object (CIO)

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

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

Oracle Configurator

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

Oracle Configurator architecture

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

Oracle Configurator Developer

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

Oracle Configurator engine

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

Oracle Configurator schema

See **CZ schema**.

Oracle Configurator Servlet

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

Oracle Configurator window

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

performance

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

Populator

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

preselection

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

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

product

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

Property

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

Property-based Compatibility Rule

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

prototype

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

PTO

Pick to Order

publication

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

publishing

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

RDBMS

Relational Database Management System

reference

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

Reference

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

Repository

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

Requires relation

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

resource

Staff or equipment available or needed within an enterprise.

Resource

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

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

reusable component

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

reusability

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

rules

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

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

runtime

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

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

schema

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

server

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

servlet

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

solution

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

SQL

Structured Query Language

Statement Rule

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

system

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

termination message

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

Total

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

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

UI

See **User Interface**.

UI Templates

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

Unknown

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

unit test

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

update

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

upgrade

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

user

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

User Interface

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

user interface

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

user requirements

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

validation

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

variable

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

verification

Tests that check whether the result agrees with the specification.

Web

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

Workbench

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

XML

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

Index

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W