Oracle® Fusion Middleware

Performance and Tuning Guide

11*g* Release 1 (11.1.1)

E10108-01

October 2009

ORACLE®

Oracle Fusion Middleware Performance and Tuning Guide 11*g* Release 1 (11.1.1)

E10108-01

Contents

Preface xv

Documentation Accessibility xv
Conventions xvi

Part I Introduction

1 Introduction and Roadmap

1.1 Document Scope and Audience 1-1
1.2 Guide to this Document 1-1
1.3 Related Documentation 1-3

2 Top Performance Areas

2.1 About Identifying Top Performance Areas 2-1
2.2 Ensure the Hardware Resources are Sufficient 2-2
2.3 Tune the Operating System 2-3
2.4 Tune Java Virtual Machines (JVMs) 2-3
2.4.1 Configuring Garbage Collection 2-4
2.4.2 Logging Low Memory Conditions 2-7
2.4.3 Monitoring and Profiling the JVM 2-7
2.5 Tune the WebLogic Server 2-8
2.6 Tune Database Parameters 2-8
2.6.1 Tuning init.ora Database Parameters 2-8
2.6.2 Tuning Redo Logs Location and Sizing 2-11
2.6.3 Automatic Segment-Space Management (ASSM) 2-11
2.7 Reuse Database Connections 2-11
2.8 Enable Data Source Statement Caching 2-12
2.9 Control Concurrency 2-12
2.9.1 HTTP Connection Limits 2-12
2.9.2 Setting the Maximum Number of Connections for Data Sources 2-14
2.9.3 Tuning the WebLogic Sever Thread Pool 2-15
2.9.4 Tuning Oracle WebCenter Concurrency 2-16
2.9.5 Tuning BPEL Concurrency 2-16
2.10 Set Logging Levels 2-16

3 Performance Planning

3.1 About Oracle Fusion Middleware Performance Planning 3-1
3.2 Performance Planning Methodology 3-1
3.2.1 Define Your Performance Objectives 3-2
3.2.2 Design Applications for Performance and Scalability 3-4
3.2.3 Monitor and Measure Your Performance Metrics 3-4

4 Monitoring Oracle Fusion Middleware

4.1 About Oracle Fusion Middleware Management Tools 4-1
4.1.1 Measuring Your Performance Metrics 4-2
4.2 Oracle Enterprise Manager 11g Fusion Middleware Control 4-2
4.2.1 Viewing Performance Metrics Using Fusion Middleware Control 4-3
4.3 Oracle WebLogic Server Administration Console 4-4
4.4 WebLogic Diagnostics Framework (WLDF) 4-5
4.5 WebLogic Scripting Tool (WLST) 4-6
4.5.1 Using Custom WLST Commands 4-7
4.6 DMS Spy Servlet 4-7
4.6.1 Viewing Performance Metrics Using the Spy Servlet 4-7
4.6.2 Using the DMS Spy Servlet 4-8
4.7 Oracle Process Manager and Notification Server 4-9
4.8 Oracle Enterprise Manager 11g Grid Control 4-9
4.9 Native Operating System Performance Commands 4-13
4.10 Network Performance Monitoring Tools 4-13

Part II Core Components

5 Oracle HTTP Server Performance Tuning

5.1 About Oracle HTTP Server 5-1
5.2 Oracle HTTP Server Directives Tuning Considerations 5-1
5.2.1 How Persistent Connections Can Reduce Httpd Process Availability 5-5
5.3 Oracle HTTP Server Logging Options 5-6
5.3.1 Access Logging 5-7
5.3.2 Configuring the HostNameLookups Directive 5-7
5.3.3 Error logging 5-7
5.4 Oracle HTTP Server Security Performance Considerations 5-7
5.4.1 Oracle HTTP Server Secure Sockets Layer (SSL) Performance Issues 5-7
5.4.2 Oracle HTTP Server Port Tunneling Performance Issues 5-10
5.5 Oracle HTTP Server Performance Tips 5-10
5.5.1 Analyze Static Versus Dynamic Requests 5-10
5.5.2 Beware of a Single Data Point Yielding Misleading Results 5-10
5.5.3 Beware of Having More Modules 5-11
5.5.4 Monitoring Oracle HTTP Server 5-11

6 Oracle Metadata Service (MDS) Performance Tuning

6.1 About Oracle Metadata Services (MDS) 6-1
6.2 Tuning Database Repository 6-1

6.2.1 Collect Schema Statistics ... 6-2
6.2.2 Increase Redo Log Size ... 6-2
6.2.3 Reclaim Disk Space ... 6-2
6.2.4 Monitor the Database Performance ... 6-2
6.3 Purging Document Version History ... 6-3
6.3.1 Auto Purge ... 6-3
6.3.2 Manual Purge ... 6-3
6.4 Using Database Polling Interval for Change Detection ... 6-3
6.5 Tuning Cache Configuration ... 6-4
6.5.1 Document Cache ... 6-5
6.6 Analyzing Performance Impact from Customization ... 6-6
6.7 Understanding DMS metrics and Characteristics ... 6-6

Part III Oracle Fusion Middleware Server Components

7 Oracle Application Development Framework Performance Tuning

7.1 About Oracle ADF ... 7-1
7.2 Oracle ADF View Performance ... 7-2
7.2.1 Oracle ADF Faces Configuration and Profiling ... 7-2
7.2.2 Performance Considerations for ADF Faces ... 7-3
7.2.3 Tuning ADF Faces Component Attributes ... 7-10
7.2.4 Performance Considerations for Table and Tree Components ... 7-12
7.2.5 Performance Considerations for autoSuggest ... 7-13
7.2.6 Data Delivery - Lazy versus Immediate ... 7-13
7.2.7 Performance Considerations for DVT Components ... 7-14
7.3 ADF Server Performance ... 7-14
7.3.1 View Objects Tuning ... 7-15
7.3.2 Batch Processing ... 7-18
7.3.3 RangeSize Tuning ... 7-18
7.3.4 Application Module Design Considerations ... 7-18
7.3.5 Application Module Pooling ... 7-19
7.3.6 ADFc: Region Usage ... 7-23
7.3.7 Reusing Static Data ... 7-23
7.3.8 Conditional Validations ... 7-23

8 Oracle TopLink (EclipseLink) JPA Performance Tuning

8.1 About Oracle TopLink and EclipseLink ... 8-1
8.2 Efficient SQL Statements and Queries ... 8-2
8.2.1 Entity Relationships Query Parameter Tuning ... 8-5
8.3 Cache Configuration Tuning ... 8-7
8.3.1 Cache Refreshing Scenarios ... 8-11
8.3.2 Locking Modes ... 8-11
8.4 Coherence Integration ... 8-13
8.5 Mapping and Descriptor Configurations ... 8-13
8.6 Analyzing EclipseLink JPA Entity Performance ... 8-13

9 Oracle Web Cache Performance Tuning

9.1 About Oracle Web Cache 9-1
9.2 Optimizing Hardware Resources 9-1
9.2.1 Hardware Resources 9-1
9.2.2 Memory Configuration 9-2
9.3 Optimizing Network Connections 9-4
9.3.1 Network Bandwidth 9-4
9.3.2 Network Connections 9-4
9.3.3 Network-Related Parameters 9-5
9.4 Optimizing Platform Connections 9-7
9.4.1 UNIX Connections 9-7
9.4.2 Windows Connections 9-7
9.5 Increasing Cache Hit Rates 9-7
9.6 Optimizing Response Time 9-9
9.7 Optimizing Performance with Oracle ADF 9-10

Part IV SOA Suite Components

10 Cross Component Tuning for SOA Suite

10.1 About SOA Suite Configuration Properties 10-1
10.2 SOA Infrastructure Configurations 10-2
10.2.1 Audit Level 10-2
10.2.2 Composite Instance State 10-2
10.2.3 Logging Level 10-3
10.3 Modifying SOA Configuration Parameters 10-3
10.4 JVM Tuning Parameters 10-3
10.5 Database Settings 10-3
10.5.1 Configuring Data Sources for SOA 10-3
10.5.2 Weblogic Server Performance Tuning 10-4

11 Oracle BPEL Process Manager Performance Tuning

11.1 About BPEL Process Manager 11-1
11.2 Basic Tuning Considerations 11-1
11.2.1 BPEL Threading Model 11-2
11.2.2 Audit Level 11-3
11.2.3 OneWayDeliveryPolicy 11-3
11.2.4 StatsLastN 11-4
11.2.5 AuditDetailThreshold 11-4
11.2.6 LargeDocumentThreshold 11-5
11.2.7 Validate XML 11-5
11.2.8 SyncMaxWaitTime 11-5
11.2.9 InstanceKeyBlockSize 11-5
11.3 BPEL Properties Set Inside a Composite 11-5
11.3.1 Component Properties 11-6
11.3.2 Partner Link Property 11-6
11.4 Tables Impacted By Instance Data Growth 11-7

12 Oracle Mediator Performance Tuning

12.1 About Oracle Mediator 12-1
12.2 Basic Tuning Considerations 12-1
12.2.1 metricsLevel 12-2
12.2.2 Domain-Value Maps 12-2
12.2.3 Deferred Routing Rules 12-2
12.2.4 Error and Retry Parameters 12-3
12.3 Event Delivery Network (EDN) Tuning 12-3

13 Oracle Human Workflow Performance Tuning

13.1 About Oracle Human Workflow 13-1
13.2 Human Workflow Tuning Considerations 13-1
13.2.1 Minimize Client Response Time 13-2
13.2.2 Choose the Right Workflow Service Client 13-2
13.2.3 Narrow Qualifying Tasks Using Precise Filters 13-2
13.2.4 Retrieve Subset of Qualifying Tasks (Paging) 13-3
13.2.5 Fetch Only the Information That Is Needed for a Qualifying Task 13-3
13.2.6 Reduce the Number of Return Query Columns 13-4
13.2.7 Use the Aggregate API for Charting Task Statistics 13-4
13.2.8 Use the Count API Methods for Counting the Number of Tasks 13-5
13.2.9 Create Indexes On Demand for Flexfields 13-5
13.2.10 Use the doesTaskExist Method 13-5
13.3 Improving Server Performance 13-5
13.3.1 Archive Completed Instances Periodically 13-6
13.3.2 Select the Appropriate Workflow Callback Functionality 13-6
13.3.3 Minimize Performance Impacts from Notification 13-6
13.3.4 Deploy Clustered Nodes 13-6
13.4 Completing Workflows Faster 13-7
13.4.1 Use Workflow Reports to Monitor Progress 13-7
13.4.2 Specify Escalation Rules 13-7
13.4.3 Specify User and Group Rules for Automated Assignment 13-7
13.4.4 Use Task Views to Prioritize Work 13-8
13.5 Tuning Identity Provider 13-8
13.6 Tuning the Database 13-8

14 Oracle Adapters Performance Tuning

14.1 About Oracle Adapters 14-1
14.2 Oracle JCA Adapters for Files/FTP 14-1
14.2.1 Inbound Throttling Best Practices 14-2
14.2.2 Outbound Throttling Best Practices 14-2
14.2.3 Outbound Performance Best Practices 14-3
14.3 Oracle JCA Adapter for Database Tuning 14-4
14.3.1 JCA Adapter Basic Tuning Considerations 14-4
14.3.2 Existence Checking 14-6
14.4 Oracle Socket Adapter Tuning 14-7
14.5 Oracle SOA JMS Adapter Tuning 14-7

14.5.1 adapter.jms.receive.threads Property 14-7
14.6 Oracle AQ Adapter Tuning 14-8
14.6.1 adapter.aq.dequeue.threads Property 14-8
14.7 Oracle MQ Adapter Tuning 14-8

15 Oracle Business Activity Monitoring Performance Tuning

15.1 About Oracle Business Activity Monitoring 15-1
15.2 Oracle BAM Tuning Considerations 15-1
15.2.1 BAM Server Tuning 15-1
15.2.2 BAM Dashboard Tuning 15-2
15.2.3 BAM Database Tuning 15-3
15.2.4 Internet Browser Tuning 15-3
15.2.5 Enterprise Message Source Tuning 15-3

16 User Messaging Service Performance Tuning

16.1 About Oracle User Messaging Services 16-1
16.2 Basic Tuning Considerations 16-1
16.2.1 SMPP Driver Performance Tuning 16-1
16.2.2 Email Driver Polling Frequency 16-2
16.3 Database Tuning for Optimal Throughput 16-2

Part V Identity Management Suite Components

17 Oracle Internet Directory Performance Tuning

17.1 About Oracle Internet Directory 17-1
17.2 Introduction to Tuning Oracle Internet Directory 17-2
17.3 Basic Tuning Recommendations 17-2
17.3.1 Database Parameters 17-2
17.3.2 LDAP Server Attributes 17-3
17.3.3 Database Statistics 17-4
17.4 Advanced Configurations 17-5
17.4.1 Replication or Oracle Directory Integration Platform 17-5
17.4.2 Replication Server Configuration 17-5
17.4.3 Garbage Collection Configuration 17-6
17.4.4 Oracle Internet Directory with RAC Database 17-7
17.4.5 Password Policies and Verifier Profiles 17-7
17.4.6 Server Entry Cache 17-8
17.4.7 Tuning Security Event Tracking 17-9
17.5 Low-Priority Tuning Recommendations 17-10
17.5.1 Number of Entries to be Returned by a Search 17-10
17.5.2 Enabling the Group Cache 17-10
17.5.3 Timeout for Write Operations 17-10
17.6 Specific Use Cases 17-11
17.6.1 Bulk Load Operation 17-11
17.6.2 Bulk Delete Operation 17-11
17.6.3 High LDAP Write Operations Load 17-11

17.7 Optimizing Searches 17-12
17.7.1 Optimizing Searches for Large Group Entries 17-12
17.7.2 Optimizing Searches for Skewed Attributes 17-13
17.7.3 Optimizing Performance of Complex Search Filters 17-13
17.8 Evaluating Performance on UNIX and Windows Systems 17-16
17.9 Obtaining Recommendations by Using the Tuning and Sizing Wizard 17-16
17.10 Updating Database Statistics by Using oidstats.sql 17-18
17.11 Setting Performance-Related Replication Configuration Attributes 17-18
17.12 Modifying Performance-Related System Configuration Attributes 17-19
17.12.1 Modifying Instance-Specific Attributes by Using Fusion Middleware Control 17-20
17.12.2 Modifying Shared Attributes by Using Fusion Middleware Control 17-20
17.12.3 Modifying Attributes by Using ldapmodify 17-21
17.13 Setting Garbage Collection Configuration Attributes 17-22
17.13.1 Modifying Changelog Purging Attributes by Using ldapmodify 17-22
17.13.2 Modifying Changelog Purging in Oracle Directory Services Manager 17-23

18 Oracle Virtual Directory Performance Tuning

18.1 About Oracle Virtual Directory 18-1
18.2 Basic Tuning Configurations 18-1
18.3 Additional Tuning Configurations 18-3
18.3.1 Database Adapters 18-3
18.3.2 Join Adapters 18-4
18.3.3 General Filter Tuning 18-4
18.3.4 Load Balancer Local Store Adapter Tuning 18-4
18.3.5 Cache Plug-In Tuning 18-4
18.3.6 LDAP Listener Tuning 18-5
18.3.7 Server Tuning 18-7

19 Oracle Identity Federation Performance Tuning

19.1 About Oracle Identity Federation 19-1
19.2 LDAP Tuning 19-1
19.2.1 Connection Pool Settings 19-2
19.2.2 Connection Settings 19-2
19.2.3 Federation Data Store Settings 19-3
19.3 Database Tuning 19-4
19.3.1 Data Sources 19-4
19.3.2 RDBMS Session Cache 19-4
19.3.3 RDBMS Compression 19-4
19.4 Oracle HTTP Server Tuning 19-5
19.5 SAML Protocol Tuning 19-5
19.5.1 SOAP Connections 19-5
19.5.2 XML Digital Signatures 19-6
19.5.3 POST and Artifact Single Sign-On Profiles 19-6

20 Oracle Fusion Middleware Security Performance Tuning

20.1 About Security Services 20-1

20.2 Detecting General Performance Issues ... 20-2
20.3 Oracle Platform Security Services Tuning ... 20-2
20.3.1 JVM Tuning Parameters ... 20-3
20.3.2 LDAP Tuning Parameters ... 20-3
20.3.3 Authentication Tuning Parameters ... 20-3
20.3.4 Authorization Tuning Parameters ... 20-3
20.3.5 OPSS Tuning Parameters for LDAP Policy Store ... 20-4
20.4 Oracle Web Services Security Tuning ... 20-5
20.4.1 Choosing the Right Policy ... 20-5
20.4.2 Timestamp (On or Off) ... 20-5
20.4.3 Policy Manager ... 20-5
20.4.4 Configuring the Log Assertion to Record SOAP Messages ... 20-6
20.4.5 Monitoring the Performance of Web Services ... 20-6

Part VI Oracle WebCenter Suite Components

21 Oracle WebCenter Performance Tuning

21.1 About Oracle WebCenter ... 21-1
21.2 Tuning WebCenter Application Configuration ... 21-2
21.3 Tuning Environment Configurations ... 21-2
21.3.1 JDBC Data Sources ... 21-2
21.3.2 Adjust Memory Size ... 21-2
21.4 Tuning Back-End Component Configuration ... 21-2
21.5 Tuning the Database ... 21-3

Part VII Capacity Planning, Scalability, and Availability

22 Capacity Planning

22.1 About Capacity Planning for Oracle Fusion Middleware ... 22-1
22.1.1 Capacity Planning Factors to Consider ... 22-2
22.2 Determining Performance Goals and Objectives ... 22-2
22.3 Measuring Your Performance Metrics ... 22-3
22.4 Identifying Bottlenecks in Your System ... 22-3
22.4.1 Using Clustered Configurations ... 22-3
22.4.2 Using Connection Pooling ... 22-4
22.4.3 Setting the Max HeapSize on JVM ... 22-4
22.4.4 Increasing Memory or CPU ... 22-4
22.4.5 Segregation of Network Traffic ... 22-4
22.4.6 Segregation of Processes and Hardware Interrupt Handlers ... 22-4
22.5 Implementing a Capacity Management Plan ... 22-5
22.5.1 Hardware Configuration Requirements ... 22-5
22.5.2 JVM Requirements ... 22-6
22.5.3 Managed Servers ... 22-6
22.5.4 Database Configuration ... 22-6

23 Using Clusters and High Availability Features

23.1 About Clusters and High Availability Features 23-1
23.2 Using Clusters with Oracle Fusion Middleware 23-2
23.3 Using High Availability Features with Oracle Fusion Middleware 23-3

Part VIII Appendixes

A Instrumenting Applications with DMS

A.1 About DMS Performance Metrics A-1
A.1.1 Instrumenting Applications with DMS A-2
A.1.2 Monitoring DMS Metrics A-2
A.1.3 Understanding DMS Terminology (Nouns and Sensors) A-3
A.1.4 DMS Naming Conventions A-6
A.2 Adding DMS Instrumentation to Java Applications A-8
A.2.1 Including DMS Imports A-9
A.2.2 Organizing Performance Data A-9
A.2.3 Defining and Using Metrics for Timing A-10
A.2.4 Defining and Using Metrics for Counting A-12
A.2.5 Defining and Using Metrics for Recording Status Information (State Sensors) A-13
A.3 Validating and Testing Applications Using DMS Metrics A-13
A.3.1 Validating DMS Metrics A-14
A.3.2 Testing DMS Metrics For Efficiency A-14
A.4 Understanding DMS Security Considerations A-15
A.5 Conditional Instrumentation Using DMS Sensor Weight A-15
A.6 Dumping DMS Metrics to Files A-16
A.7 Resetting and Destroying Sensors A-16
A.8 DMS Coding Recommendations A-16
A.8.1 Isolating Expensive Intervals Using PhaseEvent Metrics A-17
A.9 Using a High Resolution Clock to Increase DMS Precision A-18
A.9.1 Configuring DMS Clocks for Reporting Time for Java A-18
A.9.2 Configuring DMS Clocks for Reporting Time for Oracle HTTP Server A-19
A.10 Rolling Up DMS Data for Descendent Nouns A-20

B Related Reading and References

B.1 Oracle Documentation B-1
B.1.1 Oracle Fusion Middleware Library B-1
B.1.2 Oracle Database B-2
B.1.3 Oracle JRockit Java Virtual Machine (JVM) B-2
B.2 Sun Microsystems Information B-3
B.2.1 Sun Java HotSpot Virtual Machine B-3

Index

Preface

This guide describes how to monitor and optimize performance, review the key components that impact performance, use multiple components for optimal performance, and design applications for performance in the Oracle Fusion Middleware environment.

This preface contains these topics:

- Documentation Accessibility
- Conventions

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

Oracle customers have access to electronic support through My Oracle Support or by calling Oracle Support at 1.800.223.1711. Hearing-impaired customers in the U.S. who wish to speak to an Oracle Support representative may use a telecommunications relay service (TRS). Information about the TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of telephone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html. International

hearing-impaired customers should use the TRS at +1.605.224.1837. An Oracle Support engineer will respond to technical issues according to the standard service request process.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Introduction

This part describes basic performance concepts, how to measure performance, and designing applications for performance and scalability. It contains the following chapters:

- Chapter 1, "Introduction and Roadmap"
- Chapter 2, "Top Performance Areas"
- Chapter 3, "Performance Planning"
- Chapter 4, "Monitoring Oracle Fusion Middleware"

1 Introduction and Roadmap

This section describes the contents and organization of this guide.

- Section 1.1, "Document Scope and Audience"
- Section 1.2, "Guide to this Document"
- Section 1.3, "Related Documentation"

1.1 Document Scope and Audience

Oracle Fusion Middleware Performance and Tuning Guide is aimed at a target audience of Application developers, Oracle Fusion Middleware administrators, database administrators, and Web masters. This Guide assumes knowledge of Fusion Middleware Administration and hardware performance tuning fundamentals, WebLogic Server, XML, and the Java programming language.

1.2 Guide to this Document

- This chapter, Chapter 1, "Introduction and Roadmap," introduces the objectives and organization of this guide.
- Chapter 2, "Top Performance Areas," describes top tuning areas for Oracle Fusion Middleware and serves as a 'quick start' for tuning applications.
- Chapter 3, "Performance Planning," describes the performance planning methodology and tuning concepts for Oracle Fusion Middleware.
- Chapter 4, "Monitoring Oracle Fusion Middleware," describes how to monitor Oracle Fusion Middleware and its components to obtain performance data that can assist you in tuning the system and debugging applications with performance problems.
- Chapter 5, "Oracle HTTP Server Performance Tuning," discusses the techniques for optimizing Oracle HTTP Server performance, the Web server component for Oracle Fusion Middleware. It provides a listener for Oracle WebLogic Server and the framework for hosting static pages, dynamic pages, and applications over the Web.
- Chapter 6, "Oracle Metadata Service (MDS) Performance Tuning," provides tuning tips for Oracle Metadata Service (MDS). MDS is used by components such as Oracle WebCenter Framework and Oracle Application Development Framework to manage metadata.
- Chapter 7, "Oracle Application Development Framework Performance Tuning," provides basic guidelines on how to maximize the performance and scalability of

the ADF stack in applications. Oracle ADF is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications. This chapter covers design time, configuration time, and deployment time performance considerations.

- Chapter 8, "Oracle TopLink (EclipseLink) JPA Performance Tuning," provides some of the available performance options for Java Persistence API (JPA) entity architecture. Oracle TopLink includes EclipseLink as the JPA implementation.
- Chapter 9, "Oracle Web Cache Performance Tuning," provides methods and guidelines for improving the performance of Oracle Application Server Web Cache (Oracle Web Cache). Oracle Web Cache is a content-aware server accelerator or reverse proxy that improves the performance, scalability, and availability of Web sites that run on Oracle Fusion Middleware.
- Chapter 10, "Cross Component Tuning for SOA Suite," describes the common SOA infrastructure tuning parameters for configuring Oracle Service-Oriented Architecture (SOA) Suite components to improve performance. Oracle SOA Suite provides a complete set of service infrastructure components for designing, deploying, and managing SOA composite applications. Oracle SOA Suite enables services to be created, managed, and orchestrated into SOA composite applications. Composites enable you to easily assemble multiple technology components into one SOA composite application.
- Chapter 11, "Oracle BPEL Process Manager Performance Tuning," provides several BPEL property settings that can be configured to optimize performance at the process, domain, and application server levels. This chapter describes these property settings and provides recommendations on how to use them.
- Chapter 12, "Oracle Mediator Performance Tuning," describes how to tune Oracle Mediator, a service engine within the Oracle SOA Service Infrastructure, for optimal performance. Oracle Mediator provides the framework to mediate between various providers and consumers of services and events. The Mediator service engine runs with the SOA Service Infrastructure Java EE application.
- Chapter 13, "Oracle Human Workflow Performance Tuning," describes how to tune Oracle Human Workflow for optimal performance. Oracle Human Workflow is a service engine running in Oracle SOA Service Infrastructure that allows the execution of interactive human driven processes. A human workflow provides the human interaction support such as approve, reject, and reassign actions within a process or outside of any process. The Human Workflow service consists of a number of services that handle various aspects of human interaction with a business process.
- Chapter 14, "Oracle Adapters Performance Tuning," describes how to tune Oracle Adapters for optimal performance. Oracle technology adapters integrate Oracle Application Server and Oracle Fusion Middleware components such as Oracle BPEL Process Manager (Oracle BPEL PM) or Oracle Mediator components to file systems, FTP servers, database queues (advanced queues, or AQ), Java Message Services (JMS), database tables, and message queues (MQ Series).
- Chapter 15, "Oracle Business Activity Monitoring Performance Tuning," describes how to tune the Oracle Business Activity Monitoring dashboard application for optimal performance. Oracle Business Activity Monitoring (BAM) provides the tools for monitoring business services and processes in the enterprise.
- Chapter 16, "User Messaging Service Performance Tuning," describes tips for tuning the User Messaging Service. Oracle User Messaging Service (Oracle UMS) enables two way communications between users and deployed applications. It has

support for a variety of channels, such as email, IM, SMS, and text-to-voice messages. Oracle UMS is integrated with Oracle Fusion Middleware components, such as Oracle BPEL PM, Oracle Human Workflow, Oracle BAM and Oracle WebCenter.

- Chapter 17, "Oracle Internet Directory Performance Tuning," provides guidelines on Oracle Internet Directory tuning and configuration requirements. Oracle Internet Directory is an LDAP Version 3-enabled service that enables fast retrieval and centralized management of information about dispersed users, network configuration, and other resources.
- Chapter 18, "Oracle Virtual Directory Performance Tuning," provides tuning tips for Oracle Virtual Directory. Oracle Virtual Directory is an LDAP Version 3-enabled service that provides an abstracted view of one or more enterprise data sources. Oracle Virtual Directory consolidates multiple data sources into a single directory view, enabling you to integrate LDAP-aware applications with diverse directory server data stores.
- Chapter 19, "Oracle Identity Federation Performance Tuning," provides tuning tips for Oracle Identity Federation, a standalone, self-contained federation server that enables single sign-on (SSO) and authentication in a multiple-domain identity network.
- Chapter 20, "Oracle Fusion Middleware Security Performance Tuning," describes Oracle Platform Security for Java. Oracle Platform Security for Java is the Oracle Fusion Middleware security implementation for Java features such as Java Authentication and Authorization Service (JAAS) and Java EE security. This chapter describes how you can configure it for optimal performance.
- Chapter 21, "Oracle WebCenter Performance Tuning," provides suggested tuning tips for Oracle WebCenter including: Environment Configuration, Application Configuration and Back-End Services and Server Configuration.
- Chapter 22, "Capacity Planning," discusses the process of determining what type of hardware and software configuration is required to meet application needs.
- Chapter 23, "Using Clusters and High Availability Features," discusses the architecture, interaction, and dependencies of Oracle Fusion Middleware components, and explains how they can be deployed in a high availability architecture to maximize performance.
- Appendix A, "Instrumenting Applications with DMS," describes DMS and shows a sample application that demonstrates how to use DMS to instrument Java applications.
- Appendix B, "Related Reading and References," provides references to additional performance-related documentation.

1.3 Related Documentation

For more information, see the following documents in the Oracle Fusion Middleware 11*g* Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware 2 Day Administration Guide*
- *Oracle Fusion Middleware Concepts*
- *Oracle Fusion Middleware Security Guide*
- *Oracle Fusion Middleware High Availability Guide*

- *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
- *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*
- *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*

For additional documentation resources, see Appendix B, "Related Reading and References".

2

Top Performance Areas

This chapter describes the top tuning areas for Oracle Fusion Middleware. It covers critical Oracle Fusion Middleware performance areas and provides a quick start for tuning J2EE applications in the following sections:

- Section 2.1, "About Identifying Top Performance Areas"
- Section 2.2, "Ensure the Hardware Resources are Sufficient"
- Section 2.3, "Tune the Operating System"
- Section 2.4, "Tune Java Virtual Machines (JVMs)"
- Section 2.5, "Tune the WebLogic Server"
- Section 2.6, "Tune Database Parameters"
- Section 2.7, "Reuse Database Connections"
- Section 2.8, "Enable Data Source Statement Caching"
- Section 2.9, "Control Concurrency"
- Section 2.10, "Set Logging Levels"

2.1 About Identifying Top Performance Areas

One of the most challenging aspects of performance tuning is knowing where to begin. This chapter serves as a 'quick start' guide to performance tuning your Oracle Fusion Middleware applications.

Table 2–1 provides a list of common performance considerations for Oracle Fusion Middleware. While the list is a useful tool in starting your performance tuning, it is not meant to be comprehensive list of areas to tune. You must monitor and track specific performance issues within your application to understand where tuning can improve performance. See Chapter 4, "Monitoring Oracle Fusion Middleware" for more information.

Table 2–1 Top Performance Areas for Oracle Fusion Middleware

Performance Area	Description and Reference
Hardware Resources	Ensure that your hardware resources meet or exceed the application's resource requirements to maximize performance. See Section 2.2, "Ensure the Hardware Resources are Sufficient" for information on how to determine if your hardware resources are sufficient.
Operating System	Each operating system has native tools and utilities that can be useful for monitoring purposes. See Section 2.3, "Tune the Operating System"
Java Virtual Machines (JVMs)	This section discusses best practices and provides practical tips to tune the JVM and improve the performance of a J2EE application. It also discusses heap size and JVM garbage collection options. See Section 2.4, "Tune Java Virtual Machines (JVMs)".
Database	For applications that access a database, ensure that your database is properly configured to support your application's requirements. See Section 2.6, "Tune Database Parameters" for more information on garbage collection.
WebLogic Server	If your Oracle Fusion Middleware applications are using the WebLogic Server, see Section 2.5, "Tune the WebLogic Server".
Database Connections	Pooling the connections so they are reused is an important tuning consideration. See Section 2.7, "Reuse Database Connections"
Data Source Statement Caching	For applications that use a database, you can lower the performance impact of repeated statement parsing and creation by configuring statement caching properly. See Section 2.8, "Enable Data Source Statement Caching"
Oracle HTTP Server	Tune the Oracle HTTP Server directives to set the level of concurrency by specifying the number of HTTP connections. See Section 2.9, "Control Concurrency".
Concurrency	This section discusses ways to control concurrency with Oracle Fusion Middleware components. See Section 2.9, "Control Concurrency"
Logging Levels	Logging levels are thresholds that a system administrator sets to control how much information is logged. Performance can be impacted by the amount of information that Fusion applications log therefore it is important to set the logging levels appropriately. See Section 2.10, "Set Logging Levels".

2.2 Ensure the Hardware Resources are Sufficient

A key component of managing the performance of Oracle Fusion Middleware applications is to ensure that there are sufficient CPU, memory, and network resources to support the user and application requirements for your installation.

No matter how well you tune your applications, if you do not have the appropriate hardware resources, your applications cannot reach optimal performance levels. Oracle Fusion Middleware has minimum hardware requirements for its applications and database tier. For details on Oracle Fusion Middleware supported configurations, see "System Requirements and Prerequisites" in the *Oracle Fusion Middleware Installation Planning Guide* for your platform.

Sufficient hardware resources should meet or exceed the acceptable response times and throughputs for applications without becoming saturated. To verify that you have sufficient hardware resources, you should monitor resource utilization over an extended period to determine if (or when) you have occasional peaks of usage or whether a resource is consistently saturated. For more information on monitoring, see Chapter 4, "Monitoring Oracle Fusion Middleware".

> **Tip:** Your target CPU usage should not reach 100% utilization. You should determine a target CPU utilization based on your application needs, including CPU cycles for peak usage.
>
> If your CPU utilization is optimized at 100% during normal load hours, you have no capacity to handle a peak load. In applications that are latency sensitive and maintaining a fast response time is important, high CPU usage (approaching 100% utilization) can increase response times while throughput stays constant or even decreases. For such applications, a 70% - 80% CPU utilization is recommended. A good target for non-latency sensitive applications is about 90%.

If any of the hardware resources are saturated (consistently at or near 100% utilization), one or more of the following conditions may exist:

- The hardware resources are insufficient to run the application.
- The system is not properly configured.
- The application or database must be tuned.

For a consistently saturated resource, the solutions are to reduce load or increase resources. For peak traffic periods when the increased response time is not acceptable, consider increasing resources or determine if there is traffic that can be rescheduled to reduce the peak load, such as scheduling batch or background operations during slower periods.

Oracle Fusion Middleware provides a variety of mechanisms to help you control resource concurrency; this can limit the impact of bursts of traffic. However, for a consistently saturated system, these mechanisms should be viewed as temporary solutions. For more information see Section 2.9, "Control Concurrency".

2.3 Tune the Operating System

Each operating system has native tools and utilities that can be useful for monitoring and tuning purposes. Native operating system commands enable you to monitor CPU utilization, paging activity, swapping, and other system activity information.

For details on operating system commands, and guidelines for performance tuning of the network or operating system, refer to the documentation provided by the operating system vendor.

2.4 Tune Java Virtual Machines (JVMs)

How you tune your JVM greatly affects the performance of Oracle Fusion Middleware and your applications.

Note: To maximize performance from your JVM, be sure that you use only production JVMs on which your applications have been certified and that your operating system patches are up-to-date.

The Supported Configurations pages at http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html are frequently updated and contain the latest certification information on various platforms.

This section covers the following performance tuning areas for your JVM:

- Configuring Garbage Collection
- Logging Low Memory Conditions
- Monitoring and Profiling the JVM

See Also: The JVM provides a variety of parameters to enable you to more finely tune heap management and garbage collection behavior.

For more information, see the references listed in Appendix B: Oracle JRockit Java Virtual Machine (JVM) and Sun Java HotSpot Virtual Machine.

2.4.1 Configuring Garbage Collection

Garbage collection is the JVM process of freeing up unused Java objects in the Java heap. JVM garbage collection can be a resource-intensive operation and may effect application performance. In some cases, inefficient garbage collection can severely degrade application performance. Therefore, it is important to understand how applications create and destroy objects.

This section cover the following Garbage Collection tuning options:

- Specifying Heap Size Values
- Selecting a Garbage Collection Scheme
- Disabling Explicit Garbage Collection

An acceptable rate for garbage collection is application-specific and should be adjusted after analyzing the actual time and frequency of garbage collections. If you set a large heap size, full garbage collection is slower, but it occurs less frequently. If you set your heap size in accordance with your memory needs, full garbage collection is faster, but occurs more frequently.

To tune the JVM garbage collection options you must analyze garbage collection data and check for the frequency and type of garbage collections, the size of the memory pools, and the time spent on garbage collection.

Before you configure JVM garbage collection, analyze the following data points:

1. How often is garbage collection taking place? Compare the time stamps around the garbage collection.
2. How long is a full garbage collection taking?
3. What is the heap size after each full garbage collection? If the heap is always 85 percent free, for example, you might set the heap size smaller.
4. Do the young generation heap sizes (Sun) or Nursery size (Jrockit) need tuning?

You can manually log garbage collection and memory pool sizes using verbose garbage collection logging:

- Sun JVM command line options:

  ```
  -verbose:gc
  -XX:+PrintGCDetails
  -XX:+PrintGCTimeStamps
  ```

  Look for "Full GC" to identify major collections.

- Additional Sun Tools:
  - JStat
  - JConsole
  - Visualgc

  For more information on Sun's options, see http://java.sun.com/javase/technologies/hotspot/gc/index.jsp

- Jrockit JVM command line options:

  ```
  -XXverbose:gc
  ```

  NOTE: Oracle provides other command-line options to improve the performance of your JRockit VM. For detailed information, see "JRockit JDK Command Line Options by Name" at http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html

- Additional JRockit Tools:
  - JRockit Runtime Analyzer (jra recording)
  - JRockit Management Console (jrmc)
  - JRockit Memory Leak Detector

2.4.1.1 Specifying Heap Size Values

The goal of tuning your heap size is to minimize the time that your JVM spends doing garbage collection while maximizing the number of clients that the Fusion Middleware stack can handle at a given time.

Specifically the Java heap is where the objects of a Java program live. It is a repository for live objects, dead objects, and free memory. When an object can no longer be reached from any pointer in the running program, it is considered "garbage" and ready for collection. A best practice is to tune the time spent doing garbage collection to within 5% of execution time.

The JVM heap size determines how often and how long the virtual machine spends collecting garbage. An acceptable rate for garbage collection is application-specific and should be adjusted after analyzing the actual time and frequency of garbage collections. If you set a large heap size, full garbage collection is slower, but it occurs less frequently. If you set your heap size in accordance with your memory needs, full garbage collection is faster, but occurs more frequently.

In production environments, set the minimum heap size and the maximum heap size to the same value to prevent wasting virtual machine resources used to constantly grow and shrink the heap. Ensure that the sum of the maximum heap size of all the

JVMs running on your system does not exceed the amount of available physical RAM. If this value is exceeded, the Operating System starts paging and performance degrades significantly. The virtual machine always uses more memory than the heap size. The memory required for internal virtual machine functionality, native libraries outside of the virtual machine, and permanent generation memory (memory required to store classes and methods) is allocated in addition to the heap size settings.

For example, you can use the following JVM options to tune the heap:

- If you run out of heap memory (not due to a memory leak), increase `-Xmx`.
- If you run out of native memory, you may need to decrease `-Xmx`.
- For Oracle JRockit, modify `-Xns:<nursery size>` to tune the size of the nursery.
- For Sun JVM, modify `-Xmn` to tune the size of the heap for the young generation.

  If you receive java.lang.OutOfMemoryError: PermGen space errors, you may also need to increase the permanent generation space.

See Also: For more information on tuning the young generation see the "Young Generation" section of the *Java SE 6 HotSpot Virtual Machine Garbage Collection Tuning* at `http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html#generation_sizing.young_gen`

For more information on Oracle JRockit heap configurations, see "Setting the Heap and Nursery Size" in *Diagnostics Guide* at `http://download.oracle.com/docs/cd/E13188_01/jrockit/geninfo/diagnos/memman.html`

For the Sun java virtual machine see the "Insufficient Memory" section of "*Monitoring and Managing Java SE 6 Platform Applications*" at `http://java.sun.com/developer/technicalArticles/J2SE/monitoring/index.html#Insufficient_Memory`.

"Out of Memory" Frequently Asked Questions section at `http://java.sun.com/docs/hotspot/HotSpotFAQ.html#gc_oom`

2.4.1.2 Selecting a Garbage Collection Scheme

Depending on which JVM you are using, you can choose from several garbage collection schemes to manage your system memory. Some garbage collection schemes are more appropriate for a given type of application. Once you have an understanding of the workload of the application and the different garbage collection algorithms utilized by the JVM, you can optimize the configuration of the garbage collection.

Refer to the following links for garbage collection options for your JVM:

- For an overview of the garbage collection schemes available with Sun's HotSpot VM, see "Java SE 6 HotSpot Virtual Machine Garbage Collection Tuning" at `http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html`.
- For a comprehensive explanation of the collection schemes available, see "Memory Management in the Java HotSpot™ Virtual Machine" at `http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf`.

- For a discussion of the garbage collection schemes available with the JRockit JDK, see "Using the JRockit Memory Management System" at http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html.

2.4.1.3 Disabling Explicit Garbage Collection

The following parameters are used to help diagnose whether explicit garbage collections are occurring. They can also be used to disable the explicit garbage collections if necessary until the code is fixed:

- For Sun virtual machines use `-XX:+DisableExplicitGC`

  For more information on using the explicit garbage collections, see "Java SE 6 HotSpot Virtual Machine Garbage Collection Tuning " at http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html.

- For Oracle JRockit virtual machines use `-XXnoSystemGC`

  For more information on tuning the Oracle JRockit, see at http://download.oracle.com/docs/cd/E13188_01/jrockit/geninfo/diagnos/bestpractices.html

These parameters disable explicit garbage collection. Applications should avoid the use of system.gc() calls. If you suspect an application may be explicitly triggering garbage collection, set this parameter and observe the differences in your garbage collection behavior. If you detect that performance is affected by explicit collections, check the code to determine where explicit garbage collections are used and why, and the impact of disabling the calls. Application developers sometimes use system.gc() calls to trigger finalizers. This is not a recommended practice and can yield indeterminate behavior.

2.4.2 Logging Low Memory Conditions

WebLogic Server enables you to automatically log low memory conditions observed by the server. WebLogic Server detects low memory by sampling the available free memory a set number of times during a time interval. At the end of each interval, an average of the free memory is recorded and compared to the average obtained at the next interval. If the average drops by a user-configured amount after any sample interval, the server logs a low memory warning message in the log file and sets the server health state to "warning."

See Also: For more information on using WebLogic Server to detect low memory conditions refer to the following:

"Log low memory conditions" in *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

"Automatically Logging Low Memory Conditions" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*

2.4.3 Monitoring and Profiling the JVM

Monitoring the performance of your JVM is crucial to achieving optimal performance. Depending on your platform, the following tools can be used to monitor and profile your JVM:

- **Oracle JRockit® Mission Control**

Oracle JRockit Mission Control is a suite of tools designed to monitor, manage, profile, and eliminate memory leaks in your Java application without the performance impacts normally associated with these types of tools.

For more information on the Oracle JRockit Mission Control see: http://download.oracle.com/docs/cd/E13188_01/jrockit/tools/index.html

- **Sun JVM**

  The Java™ Platform comes with the following monitoring facilities built-in:

  - Java Virtual Machine Monitoring and Management API
  - JConsole
  - Hprof Tools
  - Logging Monitoring and Management Interface
  - Java Management Extensions (JMX)

  For more information on the Java platform monitoring tools, see: http://java.sun.com/developer/technicalArticles/J2SE/monitoring/

2.5 Tune the WebLogic Server

If your Oracle Fusion Middleware applications are using the WebLogic Server, see "Top Tuning Recommendations for WebLogic Server" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

2.6 Tune Database Parameters

To achieve optimal performance for applications that use the Oracle database, the database tables you access must be designed with performance in mind. Monitoring and tuning the database ensures that you get the best performance from your applications.

This section covers the following:

- Tuning init.ora Database Parameters
- Tuning Redo Logs Location and Sizing
- Automatic Segment-Space Management (ASSM)

Note: Always check the tuning guidelines in your database-specific vendor documentation. For more information on tuning the Oracle database, see the *Oracle Database Performance Tuning Guide*.

2.6.1 Tuning init.ora Database Parameters

The following tables provide common init.ora parameters and their descriptions. Consider following these guidelines to set the database parameters. Ultimately, however, the DBA should monitor the database health and tune parameters based on the need. See the following tables for more information:

- Table 2–2, " Important init.ora Oracle 10g Database Tuning Parameters"
- Table 2–3, " Important inti.ora Oracle 11g Database Tuning Parameters"

Consider applying Patch Set Release (PSR) 11.1.0.7 and upgrade the database prior to attempting the following modifications.

2.6.1.1 Initialization Parameters for Oracle 10g

The following table describes several performance-related database initialization parameters for Oracle 10*g* database.

Table 2–2 Important init.ora Oracle 10g Database Tuning Parameters

Database Parameter	Description
DB_BLOCK_SIZE	DB_BLOCK_SIZE specifies (in bytes) the size of Oracle database blocks. The default block size of 8K is optimal for most systems. Set this parameter at the time of database creation.
NLS_SORT	NLS_SORT specifies the collating sequence for ORDER BY queries. If the value is BINARY, then the collating sequence for ORDER BY queries is based on the numeric value of characters (a binary sort that requires fewer system resources). If the value is a named linguistic sort, sorting is based on the order of the defined linguistic sort. Most (but not all) languages supported by the NLS_LANGUAGE parameter also support a linguistic sort with the same name.
OPEN_CURSORS	OPEN_CURSORS specifies the maximum number of open cursors (handles to private SQL areas) a session can have at once. You can use this parameter to prevent a session from opening an excessive number of cursors. It is important to set the value of OPEN_CURSORS high enough to prevent your application from running out of open cursors. The number varies from one application to another. Assuming that a session does not open the number of cursors specified by OPEN_CURSORS, there is no added performance impact to setting this value higher than actually needed.
SESSION_CACHED_CURSORS	SESSION_CACHED_CURSORS specifies the number of session cursors to cache. Repeated parse calls of the same SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Subsequent parse calls find the cursor in the cache and do not reopen the cursor. Oracle uses a least recently used algorithm to remove entries in the session cursor cache to make room for new entries when needed. This parameter also constrains the size of the PL/SQL cursor cache which PL/SQL uses to avoid having to re-parse as statements are re-executed by a user.
SESSION_MAX_OPEN_FILES	SESSION_MAX_OPEN_FILES specifies the maximum number of BFILEs that can be opened in any session. Once this number is reached, subsequent attempts to open more files in the session by using DBMS_LOB.FILEOPEN() or OCILobFileOpen() may fail. The maximum value for this parameter depends on the equivalent parameter defined for the underlying operating system.
JOB_QUEUE_PROCESSES	JOB_QUEUE_PROCESSES specifies the maximum number of processes that can be created for the execution of jobs. It specifies the number of job queue processes per instance.
LOG_BUFFER	LOG_BUFFER specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. The LGWR process writes redo log entries from the log buffer to a redo log file.
UNDO_MANAGEMENT	UNDO_MANAGEMENT specifies which undo space management mode the system should use. When set to AUTO, the instance starts in automatic undo management mode. In manual undo management mode, undo space is allocated externally as rollback segments.

Table 2–2 (Cont.) Important init.ora Oracle 10g Database Tuning Parameters

Database Parameter	Description
PL_SQL_CODE_TYPE	PLSQL_CODE_TYPE specifies the compilation mode for PL/SQL library units. INTERPRETED: PL/SQL library units are compiled to PL/SQL bytecode format. Such modules are executed by the PL/SQL interpreter engine. NATIVE: PL/SQL library units are compiled to native (machine) code. Such modules are executed natively without incurring any interpreter impacts.
PROCESSES	Sets the maximum number of operating system processes that can be connected to Oracle concurrently. The value of this parameter must account for Oracle background processes. SESSIONS parameter is deduced from this value.
PGA_AGGREGATE_TARGET	Specifies the target aggregate PGA memory available to all server processes attached to the instance.
SGA_MAX_SIZE	This parameter is the maximum size of the SGA for a running instance. Set this parameter to the amount of memory that you want dedicated for the SGA, which includes the following memory pools: ■ Database buffer cache ■ Shared pool ■ Large pool ■ Java pool Ensure that you regularly monitor the buffer cache hit ratio and size the SGA so that the buffer cache has an adequate number of frames for the workload. The buffer cache hit ratio may be calculated from data in the view `V$SYSSTAT`. Also the view `V$DB_CACHE_ADVICE` provides data that can be used to tune the buffer cache.
SGA_TARGET	Setting this parameter to a nonzero value enables Automatic Shared Memory Management. Consider using automatic memory management, both to simplify configuration and to improve performance.
TRACE_ENABLED	TRACE_ENABLED controls tracing of the execution history, or code path, of Oracle. Oracle Support Services uses this information for debugging. Although the performance impacts incurred from processing is not excessive, you may improve performance by setting TRACE_ENABLED to FALSE.

2.6.1.2 Initialization Parameters for Oracle 11g

The following table provides information on some important performance-related database initialization parameters for Oracle 11g database.

Table 2–3 Important inti.ora Oracle 11g Database Tuning Parameters

Database Parameter	Description
AUDIT_TRAIL	AUDIT_TRAIL enables or disables database auditing.
MEMORY_MAX_TARGET	MEMORY_MAX_TARGET specifies the maximum value to which a DBA can set the MEMORY_TARGET initialization parameter.
MEMORY_TARGET	MEMORY_TARGET specifies the Oracle system-wide usable memory. The database tunes memory to the MEMORY_TARGET value, reducing or enlarging the SGA and PGA as needed.
PGA_AGGREGATE_TARGET	Specifies the target aggregate PGA memory available to all server processes attached to the instance. In Oracle 11*g*, set MEMORY_TARGET instead of setting SGA and the PGA separately.
SGA_MAX_SIZE	Consider setting MEMORY_TARGET instead of setting SGA and the PGA separately.
SGA_TARGET	Consider setting MEMORY_TARGET instead of setting SGA and the PGA separately.

2.6.2 Tuning Redo Logs Location and Sizing

Managing the database I/O load balancing is a non-trivial task. However, tuning the redo log options can provide performance improvement for applications running in an Oracle Fusion Middleware environment, and in some cases, you can significantly improve I/O throughput by moving the redo logs to a separate disk.

The size of the redo log files can also influence performance, because the behavior of the database writer and archiver processes depend on the redo log sizes. Generally, larger redo log files provide better performance by reducing checkpoint activity. It is not possible to provide a specific size recommendation for redo log files, but redo log files in the range of a hundred megabytes to a few gigabytes are considered reasonable. Size your online redo log files according to the amount of redo your system generates. A rough guide is to switch logs at most once every twenty minutes. Set the initialization parameter `LOG_CHECKPOINTS_TO_ALERT = TRUE` to have checkpoint times written to the alert file.

The complete set of required redo log files can be created during database creation. After they are created, the size of a redo log size cannot be changed. New, larger files can be added later, however, and the original (smaller) ones can be dropped. For more information see the *Oracle Database Performance Tuning Guide*.

2.6.3 Automatic Segment-Space Management (ASSM)

For permanent tablespaces, consider using automatic segment-space management. Such tablespaces, often referred to as bitmap tablespaces, are locally managed tablespaces with bitmap segment space management.

For backward compatibility, the default local tablespace segment-space management mode is `MANUAL`.

For more information, see "Free Space Management" in *Oracle Database Concepts*, and "Specifying Segment Space Management in Locally Managed Tablespaces" in *Oracle Database Administrator's Guide*.

2.7 Reuse Database Connections

Creating a database connection is a relatively resource intensive process in any environment. Typically, a connection pool starts with a small number of connections. As client demand for more connections grow, there may not be enough in the pool to satisfy the requests. WebLogic Server creates additional connections and adds them to the pool until the maximum pool size is reached.

One way to avoid connection creation delays is to initialize all connections at server startup, rather than on-demand as clients need them. This may be appropriate if your load is predictable and even. Set the initial number of connections equal to the maximum number of connections in the Connection Pool tab of your data source configuration. Determine the optimal value for the Maximum Capacity as part of your pre-production performance testing.

If your load is uneven, and has a much higher number of connections at peak load than at typical load, consider setting the initial number of connections equal to your typical load. In addition, consider setting the maximum number of connections based on your supported peak load. With these configurations, WebLogic server can free up some connections when they are not used for a period of time.

For more information, see "Tuning Data Source Connection Pool Options" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

2.8 Enable Data Source Statement Caching

When you use a prepared statement or callable statement in an application or EJB, there may be a performance impact associated with the processing of the communication between the application server and the database server and on the database server. To minimize the processing impact, enable the data source to cache prepared and callable statements used in your applications. When an application or EJB calls any of the statements stored in the cache, the server reuses the statement stored in the cache. Reusing prepared and callable statements reduces CPU usage on the database server, improving performance for the current statement and leaving CPU cycles for other tasks.

Each connection in a data source has its own individual cache of prepared and callable statements used on the connection. However, you configure statement cache options per data source. That is, the statement cache for each connection in a data source uses the statement cache options specified for the data source, but each connection caches it's own statements. Statement cache configuration options include:

- Statement Cache Type—The algorithm that determines which statements to store in the statement cache.
- Statement Cache Size—The number of statements to store in the cache for each connection. The default value is 10. You should analyze your database's statement parse metrics to size the statement cache sufficiently for the number of statements you have in your application.

You can use the Administration Console to set statement cache options for a data source. See "Configure the statement cache for a JDBC data source" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

For more information on using statement caching, see "Increasing Performance with the Statement Cache" in the *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

2.9 Control Concurrency

Limiting concurrency, at multiple layers of the system to match specific usage needs, can greatly improve performance. This section discusses a few of the areas within Oracle Fusion Middleware where concurrency can be controlled.

When system capacity is reached, and a web server or application server continues to accept requests, application performance and stability can deteriorate. There are several places within Oracle Fusion Middleware where you can throttle the requests to avoid overloading the mid-tier or database tier systems and tune for best performance.

- HTTP Connection Limits
- Setting the Maximum Number of Connections for Data Sources
- Tuning the WebLogic Sever Thread Pool
- Tuning Oracle WebCenter Concurrency
- Tuning BPEL Concurrency

2.9.1 HTTP Connection Limits

Oracle HTTP Server uses directives in `httpd.conf`. This configuration file specifies the maximum number of HTTP requests that can be processed simultaneously, logging details, and certain limits and time outs.

For more information on modifying the httpd.conf file, see "Configuring Oracle HTTP Server" in *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

You can use the `MaxClients` and `ThreadsPerChild` directives to limit incoming requests to WebLogic instances from the Oracle HTTP Server based on your expected client load and system resources. The following sections describe some Oracle HTTP Server tuning parameters related to connection limits that you typically need to tune based on your expected client load. See Chapter 5, "Oracle HTTP Server Performance Tuning" for more information and a more complete list of tunable parameters.

2.9.1.1 MaxClients/ThreadsPerChild

> **Note:** The `MaxClients` parameter is applicable only to UNIX platforms and on Microsoft Windows (mpm_winnt), the same is achieved through the `ThreadsPerChild` and `ThreadLimit` parameters.

The `MaxClients` property specifies a limit on the total number of server threads running, that is, a limit on the number of clients who can simultaneously connect. If the number of client connections reaches this limit, then subsequent requests are queued in the TCP/IP system up to the limit specified (in the ListenBackLog directive).

You can configure the `MaxClients` directive in the httpd.conf file up to a maximum of 8K (the default value is 150). If your system is not resource-saturated and you have a user population of more than 150 concurrent HTTP connections, you can improve your performance by increasing `MaxClients` to increase server concurrency. Increase `MaxClients` until your system becomes fully utilized (85% is a good threshold).

When system resources are saturated, increasing `MaxClients` does not improve performance. In this case, the `MaxClients` value could be reduced as a throttle on the number of concurrent requests on the server.

If the server handles persistent connections, then it may require sufficient concurrent httpd server processes to handle both active and idle connections. When you specify `MaxClients` to act as a throttle for system concurrency, you need to consider that persistent idle httpd connections also consume httpd processes. Specifically, the number of connections includes the currently active persistent and non-persistent connections and the idle persistent connections. When there are no httpd server threads available, connection requests are queued in the TCP/IP system until a thread becomes available, and eventually clients terminate connections.

You can define a number of server processes and the threads per process (`ThreadsPerChild`) to handle the incoming connections to Oracle HTTP Server. The `ThreadsPerChild` property specifies the upper limit on the number of threads that can be created under a server (child) process.

> **Note:** `ThreadsPerChild`, `StartServers`, and `ServerLimit` properties are inter-related with the `MaxClients` setting. All of these properties must be set appropriately to achieve the number of connections as specified by `MaxClients`. See Table 5–1, " Oracle HTTP Server Configuration Properties" for a description of all the HTTP configuration properties.

2.9.1.2 KeepAlive

A persistent, `KeepAlive`, HTTP connection consumes an httpd child process, or thread, for the duration of the connection, even if no requests are currently being processed for the connection.

If you have sufficient capacity, `KeepAlive` should be enabled; using persistent connections improves performance and prevents wasting CPU resources re-establishing HTTP connections. Normally, you should not need to change `KeepAlive` parameters.

Note: The default maximum requests for a persistent connection is 100, as specified with the `MaxKeepAliveRequests` directive in httpd.conf. By default, the server waits for 15 seconds between requests from a client before closing a connection, as specified with the `KeepAliveTimeout` directive in httpd.conf.

2.9.1.3 Tuning MOD_WL_OHS

The Oracle HTTP Server (OHS) uses the mod_wl_ohs module to route requests to the underlying Weblogic server or the Weblogic Server cluster. The configuration details for mod_wl_ohs are available in the `mod_wl_ohs.conf` file in the config directory.

For more information on the tuning parameters for mod_wl_ohs see, "Understanding Oracle HTTP Server Modules" in *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

2.9.2 Setting the Maximum Number of Connections for Data Sources

For applications that use a database, performance can improve when the connection pool associated with a data source limits the number of connections. You can use the `MaxCapacity` attribute to limit the database requests from Oracle Application Server so that incoming requests do not saturate the database, or to limit the database requests so that the database access does not overload the Oracle Application Server-tier resource.

The connection pool `MaxCapacity` attribute specifies the maximum number of connections that a connection pool allows. By default, the value of `MaxCapacity` is set to 15. For best performance, you should specify a value for `MaxCapacity` that matches the number appropriate to your database performance characteristics.

Limiting the total number of open database connections to a number your database can handle is an important tuning consideration. You should check to make sure that your database is configured to allow at least as large a number of open connections as the total of the values specified for all the data sources `MaxCapacity` option, as specified in all the applications that access the database.

See Also: "JDBC Data Source: Configuration: Connection Pool" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

"Tuning Data Source Connection Pool Options" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

2.9.3 Tuning the WebLogic Sever Thread Pool

By default WebLogic Server uses a single thread pool, in which all types of work are executed. WebLogic Server uses Work Managers to prioritize work based on rules you can define, and run-time metrics, including the actual time it takes to execute a request and the rate at which requests are entering and leaving the pool. There is a default work manager that manages the common thread pool.

The common thread pool changes its size automatically to maximize throughput. WebLogic Server monitors throughput over time and based on history, determines whether to adjust the thread count. For example, if historical throughput statistics indicate that a higher thread count increased throughput, WebLogic increases the thread count. Similarly, if statistics indicate that fewer threads did not reduce throughput, WebLogic decreases the thread count.

Since the WebLogic Server thread pool by default is sized automatically, in most situations you do not need to tune this. However, for special requirements, an administrator can configure custom Work Managers to manage the thread pool at a more granular level for sets of requests that have similar performance, availability, or reliability requirements. With custom work managers, you can define priorities and guidelines for how to assign pending work (including specifying a min threads or max threads constraint, or a constraint on the total number of requests that can be queued or executing before WebLogic Server begins rejecting requests).

Use the following guidelines to help you determine when to use Work Managers to customize thread management:

- The default fair share is not sufficient.

  This usually occurs in situations where one application needs to be given a higher priority over another.

- A response time goal is required.
- A minimum thread constraint needs to be specified to avoid server deadlock.
- You use MDBs in your application.

  To ensure MDBs use a well-defined share of server thread resources, and to tune MDB concurrency, most MDBs should be modified to reference a custom work manager that has a max-threads-constraint. In general, a custom work manager is useful when you have multiple MDB deployments, or if you determine that a particular MDB needs more threads.

See Also: For more information on how to use custom Work Managers to customize thread management, and when to use custom work managers, see the following:

- "Tune Pool Sizes" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
- "Thread Management" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
- "MDB Thread Management" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
- "Using Work Managers to Optimize Scheduled Work" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*
- "Avoiding and Managing Overload" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*

You can use Oracle WebLogic Administration Console to view general information about the status of the thread pool (such as active thread count, total thread count, and queue length.) You can also use the Console to view each application's scoped work manager metrics from the Workload tab on the Monitoring page. The metrics provided include the number of pending requests and number of completed requests.

For more information, see "Servers: Monitoring: Threads" and "Deployments: Monitoring: Workload" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

The work manager and thread pool metrics can also be viewed from the Oracle Fusion Middleware Control. For more information, see Section 4.2.1, "Viewing Performance Metrics Using Fusion Middleware Control".

2.9.4 Tuning Oracle WebCenter Concurrency

Oracle WebCenter has its own controls for managing concurrency. See "Configuring Concurrency Management" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

2.9.5 Tuning BPEL Concurrency

The Oracle BPEL Process Manager has its own thread controls and specialized tuning. See Section 11.2.1, "BPEL Threading Model".

2.10 Set Logging Levels

The amount of information that Fusion applications log depends on how the environment is configured and how the application code is instrumented. To maximize performance it is recommended that the logging level is not set higher than the default INFO level logging. If the logging setting does not match the default level, reset the logging level to the default for best performance.

Once the application and server logging levels are set appropriately, ensure that the debugging properties or other application level debugging flags are set to appropriate

levels or disabled. To avoid performance impacts, do not set log levels to levels that produce more diagnostic messages, including the `FINE` or `TRACE` levels.

For more information see setting appropriate logging levels for your Oracle Fusion Middleware applications, see "Configuring Your Development Environment for Logging and Diagnostics" in *Oracle Fusion Applications Developer's Guide*.

3 Performance Planning

This chapter discusses performance and tuning concepts for Oracle Fusion Middleware. This chapter contains the following sections:

- Section 3.1, "About Oracle Fusion Middleware Performance Planning"
- Section 3.2, "Performance Planning Methodology"

3.1 About Oracle Fusion Middleware Performance Planning

To maximize Oracle Fusion Middleware performance, you must monitor, analyze, and tune all the components that are used by your applications. This guide describes the tools that you can use to monitor performance and the techniques for optimizing the performance of Oracle Fusion Middleware components.

Performance tuning usually involves a series of trade-offs. After you have determined what is causing the bottlenecks, you may have to modify performance in some other areas to achieve the expected results. However, if you have a clearly defined plan for achieving your performance objectives, the decision on what to trade for higher performance is easier because you have identified the most important areas.

If you are new to Oracle Fusion Middleware, or if you would like more information on the Oracle Fusion Middleware components, refer to documentation listed in Appendix B, "Related Reading and References".

3.2 Performance Planning Methodology

The Fusion Middleware components are built for performance and scalability. To maximize the performance capabilities of your applications, you must build performance and scalability into your design. The performance plan should address the current performance requirements, the existing issues (such as bottlenecks or insufficient hardware resources) and any anticipated variances in load, users or processes. The performance plan should also address how the components scale during peak usage without impacting performance.

The following sections of this chapter discuss the steps you should take to help create a plan to tune your application environment and optimize performance:

- Step 1: Define Your Performance Objectives
- Step 2: Design Applications for Performance and Scalability
- Step 3: Monitor and Measure Your Performance Metrics

3.2.1 Define Your Performance Objectives

Before you can begin performance tuning your applications, you must first identify the performance objectives you hope to achieve. To determine your performance objectives, you must understand the applications deployed and the environmental constraints placed on the system.

To understand what your performance objectives are, you must complete the following steps:

- Define Operational Requirements
- Identify Performance Goals
- Understand User Expectations
- Conduct Performance Evaluations

Performance objectives are limited by constraints, such as:

- The configuration of hardware and software such as CPU type, disk size, disk speed, and sufficient memory.

  There is no single formula for determining your hardware requirements. The process of determining what type of hardware and software configuration is required to meet application needs adequately is called *capacity planning*.

  Capacity planning requires assessment of your system performance goals and an understanding of your application. Capacity planning for server hardware should focus on maximum performance requirements. For more information on capacity planning, see Chapter 22, "Capacity Planning".
- The configuration of high availability architecture to address peak usage and response times. For more information on implementing high availability features in Oracle Fusion Middleware applications, see Chapter 23, "Using Clusters and High Availability Features".
- The ability to interoperate between domains, use legacy systems, support legacy data.
- Development, implementation, and maintenance costs.

Understanding these constraints - and their impacts - ensure that you set realistic performance objectives for your application environment, such as response times, throughput, and load on specific hardware.

3.2.1.1 Define Operational Requirements

Before you begin to deploy and tune your application on Oracle Fusion Middleware, it is important to clearly define the operational environment. The operational environment is determined by high-level constraints and requirements such as:

- Application Architecture
- Security Requirements
- Hardware Resources

3.2.1.2 Identify Performance Goals

Whether you are designing a new system or maintaining an existing system, you should set specific performance goals so that you know how and what to optimize. To determine your performance objectives, you must understand the application deployed and the environmental constraints placed on the system.

Gather information about the levels of activity that components of the application are expected to meet, such as:

- Anticipated number of users
- Number and size of requests
- Amount of data and its consistency
- Target CPU utilization

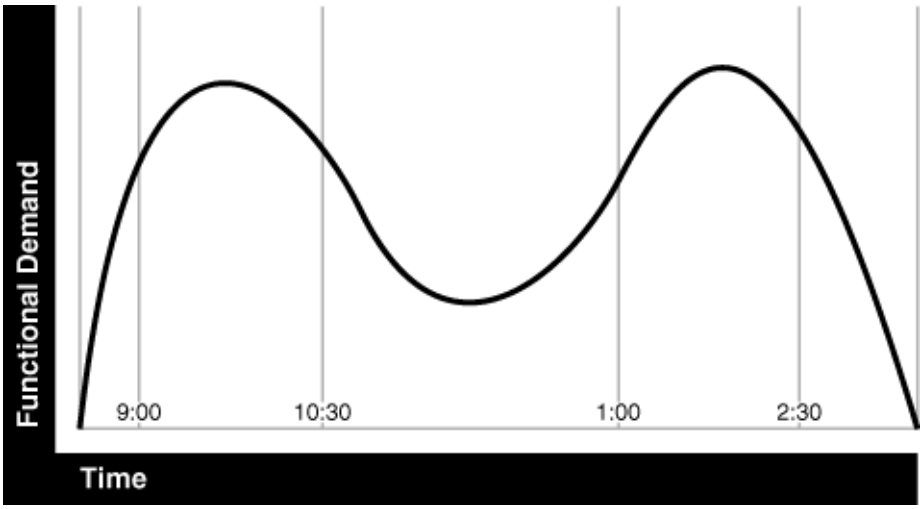
3.2.1.3 Understand User Expectations

Application developers, database administrators, and system administrators must be careful to set appropriate performance expectations for users. When the system carries out a particularly complicated operation, response time may be slower than when it is performing a simple operation. Users should be made aware of which operations might take longer.

For example, you might want to ensure that 90% of the users experience response times no greater than 5 seconds and the maximum response time for all users is 20 seconds. Usually, it's not that simple. Your application may include a variety of operations with differing characteristics and acceptable response times. You need to set measurable goals for each of these.

You also need to determine how variances in the load can affect the response time. For example, users might access the system heavily between 9:00am and 10:00am and then again between 1:00pm and 2:00pm, as illustrated by the graph in Figure 3–1. If your peak load occurs on a regular basis, for example, daily or weekly, the conventional wisdom is to configure and tune systems to meet your peak load requirements. The lucky users who access the application in off-time can experience better response times than your peak-time users. If your peak load is infrequent, you may be willing to tolerate higher response times at peak loads for the cost savings of smaller hardware configurations.

Figure 3–1 Adjusting Capacity and Functional Demand



3.2.1.4 Conduct Performance Evaluations

With clearly defined performance goals and performance expectations, you can readily determine when performance tuning has been successful. Success depends on the

functional objectives you have established with the user community, your ability to measure whether the criteria are being met, and your ability to take corrective action to overcome any exceptions.

Ongoing performance monitoring enables you to maintain a well-tuned system. Keeping a history of the application's performance over time enables you to make useful comparisons. With data about actual resource consumption for a range of loads, you can conduct objective scalability studies and from these predict the resource requirements for anticipated load volumes. For more information on evaluating performance, see Chapter 4, "Monitoring Oracle Fusion Middleware".

3.2.2 Design Applications for Performance and Scalability

The key to good performance is good design. The design phase of the application development cycle should be an on-going process. Cycling through the planning, monitoring and tuning phases of the application development cycle is critical to achieving optimal performance across Fusion Middleware deployments. Using an iterative design methodology enables you to accommodate changes in your work loads without impacting your performance objectives.

See the following Oracle Fusion Middleware developer's documentation for more information on recommended design techniques:

- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*
- *Oracle Fusion Middleware Developer's Guide for Oracle TopLink*
- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Application Developer's Guide for Oracle Identity Management*

3.2.3 Monitor and Measure Your Performance Metrics

Oracle Fusion Middleware provides a variety of technologies and tools that can be used to monitor Server and Application performance. Monitoring enables you to evaluate Server activity, watch trends, diagnose system bottlenecks, debug applications with performance problems and gather data that can assist you in tuning the system. For more information, see Chapter 4, "Monitoring Oracle Fusion Middleware.".

Performance tuning is specific to the applications and resources that you have deployed on your system. Some common tuning areas are included in Chapter 2, "Top Performance Areas."

See Also: *Oracle Database Performance Tuning Guide*

Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server

Oracle Fusion Middleware Administrator's Guide

4

Monitoring Oracle Fusion Middleware

Oracle Fusion Middleware provides a variety of technologies and tools that can be used to monitor Server and Application performance. Monitoring is an important step in performance tuning and enables you to evaluate server activity, watch trends, diagnose system bottlenecks, debug applications with performance problems and gather data that can assist you in tuning the system.

This chapter contains the following sections:

- Section 4.1, "About Oracle Fusion Middleware Management Tools"
- Section 4.2, "Oracle Enterprise Manager 11g Fusion Middleware Control"
- Section 4.3, "Oracle WebLogic Server Administration Console"
- Section 4.4, "WebLogic Diagnostics Framework (WLDF)"
- Section 4.5, "WebLogic Scripting Tool (WLST)"
- Section 4.6, "DMS Spy Servlet"
- Section 4.7, "Oracle Process Manager and Notification Server"
- Section 4.8, "Oracle Enterprise Manager 11g Grid Control"
- Section 4.9, "Native Operating System Performance Commands"
- Section 4.10, "Network Performance Monitoring Tools"

4.1 About Oracle Fusion Middleware Management Tools

After you install and configure Oracle Fusion Middleware, you can use the graphical user interfaces or command-line tools to manage your environment.

You can use the following tools to manage your Oracle Fusion Middleware installations:

- Oracle Enterprise Manager Fusion Middleware Control. See Section 4.2.
- Oracle WebLogic Server Administration Console. See Section 4.3.
- Oracle WebLogic Diagnostics Framework (WLDF). See Section 4.4.
- Oracle WebLogic Scripting Tool (WLST). See Section 4.5.
- DMS Spy Servlet. See Section 4.6.
- Oracle Process Manager and Notification Server. See Section 4.7.
- Oracle Enterprise Manager 11g Grid Control. See Section 4.8.
- Operating System Performance Commands. See Section 4.9.

- Network Performance Monitoring Tools. See Section 4.10.

Use these tools, rather than directly editing configuration files, to perform all administrative tasks unless a specific procedure requires you to edit a file. Editing a file may cause the settings to be inconsistent and generate problems.

Both Fusion Middleware Control and Oracle WebLogic Server Administration Console are graphical user interfaces that you can use to monitor and administer your Oracle Fusion Middleware environment. You can perform some tasks with either tool, but, for other tasks, you can only use one of the tools.

For more information on using WebLogic Server Administration Console for monitoring your domain, see the *Oracle Fusion Middleware Administrator's Guide*.

4.1.1 Measuring Your Performance Metrics

Metrics are the criteria you use to measure your scenarios against your performance objectives. You can use performance metrics to help locate bottlenecks, identify resource availability issues, or help tune your components to improve throughput and response times. After you have determined your performance criteria, take measurements of the metrics used to quantify your performance objectives.

For example, you might use response time, throughput, and resource utilization as your metrics. The performance objective for each metric is the value that is acceptable. You match the actual value of the metrics to your objectives to verify that you are meeting, exceeding, or failing to meet your performance objectives.

When you manage or monitor an Oracle Fusion Middleware component or application with Fusion Middleware Control, you may see performance metrics that provide insight into the current performance of the component or application. In many cases, these metrics are shown in interactive charts; other times they are presented in tabular format. The best way to use and correlate the performance metrics is from the Performance Summary page for the component or application you are monitoring.

The next sections of this chapter provide an overview of the Oracle Fusion Middleware technologies and tools that can be used to monitor Server and Application performance. If you are new to Oracle Fusion Middleware or if you need additional information about monitoring your environment using the Performance Summary pages, see "Viewing the Performance of Oracle Fusion Middleware" in the *Oracle Fusion Middleware Administrator's Guide*. In addition, the Fusion Middleware Control online help provides definitions and other information about specific performance metrics that are available on its management and monitoring pages. See Section 4.2.1, "Viewing Performance Metrics Using Fusion Middleware Control".

4.2 Oracle Enterprise Manager 11g Fusion Middleware Control

Fusion Middleware Control is a Web browser-based, graphical user interface that you can use to monitor and administer a farm. Fusion Middleware Control organizes a wide variety of performance data and administrative functions into distinct, Web-based home pages for the farm, domain, servers, components, and applications. The Fusion Middleware Control home pages make it easy to locate the most important monitoring data and the most commonly used administrative functions—all from your Web browser.

In addition, Fusion Middleware Control provides a set of MBean browsers that allow you to browse the MBeans for a WebLogic Server or for a selected application and perform specific monitoring and configuration tasks from the MBean browser.

> **See Also:** "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide*

Use Fusion Middleware Control to:

- Monitor and administer a single Fusion Middleware Farm
- Monitor all elements of the farm - including deployed applications and Fusion Middleware components such as:
  - WebLogic Domain
  - Cluster and Managed Servers
  - SOA components
  - Web Center
  - Web Cache
  - Oracle HTTP Server
  - Oracle Identity Management
- Monitor the state and performance of each of these targets by providing out-of-the-box performance metrics
- Monitor CPU usage, heap usage, Work Manager, JMS servers, and JDBC and JTA usage for Oracle WebLogic Server
- Monitor JVM performance in terms of heap versus non-heap usage, garbage collection, and threads performance
- Monitor applications and Web services deployed to WebLogic Server
- Monitor a wide range of application metrics for servlets, JSPs, and EJBs are available, as well as Web services metrics for faults, invocations, and violations. Such metrics are accessible from a target's home page.
- Access customizable performance summary pages to help administrators monitor performance and diagnose problems. These charts can be modified to display content that is relevant to your domain. A target or component might be added to the chart so that you can compare the performance information for two targets in one chart.

> **See Also:** For more information about monitoring your environment using the Performance Summary pages, see "Viewing the Performance of Oracle Fusion Middleware" in *Oracle Fusion Middleware Administrator's Guide*.

4.2.1 Viewing Performance Metrics Using Fusion Middleware Control

When you manage or monitor an Oracle Fusion Middleware component or application with Fusion Middleware Control, you often see performance metrics that provide insight into the current performance of the component or application. In many cases, these metrics are shown in interactive charts; other times they are presented in tabular format. The best way to use and correlate the performance metrics is from the Performance Summary page for the component or application you are monitoring.

Use the Fusion Middleware Control online help to obtain a definition of a specific performance metric. There are two ways to access this information:

- Browse or search for the metric in the Fusion Middleware Control online help.
- Navigate to the Performance Summary page for your Oracle Fusion Middleware component or application and do the following:
  1. Click **Show Metric Palette**.
  2. Browse the list of metrics available for the component or application to locate a specific metric.
  3. Right-click the name of the metric and select **Help** from the context menu.

If you encounter a problem, such as an application that is running slowly or is hanging, you can view more detailed performance information, including performance metrics for a particular target, to find out more information about the problem.

Oracle Fusion Middleware automatically and continuously measures run-time performance. The performance metrics are automatically enabled; you do not need to set options or perform any extra configuration to collect them. If you are interested in viewing historical data, consider using Oracle Enterprise Manager Grid Control. For more information see "Middleware Management" in *Oracle Enterprise Manager Concepts*.

4.3 Oracle WebLogic Server Administration Console

Oracle WebLogic Server Administration Console is a Web browser-based, graphical user interface that you use to manage an Oracle WebLogic Server domain. It is accessible from any supported Web browser with network access to the Administration Server.

> **See Also:** For general information on using the WebLogic Server console, see "Getting Started Using Oracle WebLogic Server Administration Console" in *Oracle Fusion Middleware Administrator's Guide*.

Use the WebLogic Server Administration Console to:

- Configure, start, and stop WebLogic Server instances
- Configure and Monitor WebLogic Server clusters
- Configure and Monitor WebLogic Server services, such as database connectivity (JDBC) and messaging (JMS)
- Configure security parameters, including creating and managing users, groups, and roles
- Configure and deploy Java EE applications
- Monitor server and application performance
- View server and domain log files
- View application deployment descriptors
- Edit selected run-time application deployment descriptor elements

Oracle WebLogic Server contains a Java Management Extensions (JMX) server implementation and provides its own set of Management Beans (MBeans). Oracle management tools described in this chapter use the MBeans provided by WebLogic Server to allow you to configure, monitor, and manage WebLogic Server resources.

Additional WebLogic Server Console Resources:

For details on the content contained in each summary table, see "Monitor Servers" in WebLogic Administration Console Online Help.

For detailed information on using the WebLogic Server to monitor your domain, see the *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

The Oracle Technology Network at `http://www.oracle.com/technology/index.html` provides product downloads, articles, sample code, product documentation, tutorials, white papers, news groups, and other key content for WebLogic Server.

4.4 WebLogic Diagnostics Framework (WLDF)

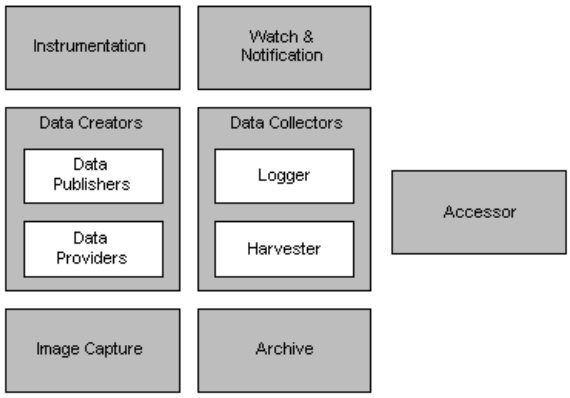
The WebLogic Diagnostic Framework (WLDF) is a monitoring and diagnostic framework that can collect diagnostic data that servers and applications generate. The WLDF can be configured to collect the data and store it in various sources, including log records, data events, and harvested metrics.

WLDF includes several components for collecting and analyzing data:

- **Data Creator**s— data publishers and data providers that are distributed across WLDF components.
- **Diagnostic Image Capture**—Creates a diagnostic snapshot from the server that can be used for post-failure analysis.
- **Archive**—Captures and persists data events, log records, and metrics from server instances and applications.
- **Instrumentation**—Adds diagnostic code to WebLogic Server instances and the applications running on them to execute diagnostic actions at specified locations in the code. The Instrumentation component provides the means for associating a diagnostic context with requests so they can be tracked as they flow through the system.
- **Harvester**—Captures metrics from run-time MBeans, including WebLogic Server MBeans and custom MBeans, which can be archived and later accessed for viewing historical data.
- **Watches and Notifications**—Provides the means for monitoring server and application states and sending notifications based on criteria set in the watches. (A watch rule can monitor log data, event data from the Instrumentation component, or metric data from a data provider that is harvested by the Harvester. The Watch Manager is capable of managing watches that are composed of several watch rules.)
- **Logging services**—Manage logs for monitoring server, subsystem, and application events.

The relationship among these components is shown in Figure 4–1.

Figure 4–1 Major WLDF Components



All of the framework components operate at the server level and are only aware of server scope. All the components exist entirely within the server process and participate in the standard server lifecycle. All artifacts of the framework are configured and stored on a per server basis.

> **Note:** For more information on the WebLogic Diagnostics Framework and how it can be leveraged for monitoring Oracle Fusion Middleware components, see *Oracle Fusion Middleware Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

4.5 WebLogic Scripting Tool (WLST)

The Oracle WebLogic Scripting Tool (WLST) is a command-line scripting environment that you can use to create, manage, and monitor Oracle WebLogic Server domains. It is based on the Java scripting interpreter, Jython. In addition to supporting standard Jython features such as local variables, conditional variables, and flow control statements, WLST provides a set of scripting functions (commands) that are specific to WebLogic Server. You can extend the WebLogic scripting language to suit your needs by following the Jython language syntax.

You can use any of the following techniques to invoke WLST commands:

- Interactively, on the command line
- In script mode, supplied in a file
- Embedded in Java code

> **See Also:**
>
> - *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*
> - "Using Custom WLST Commands" in *Oracle Fusion Middleware Administrator's Guide*

4.5.1 Using Custom WLST Commands

Many components, such as Oracle SOA Suite, Oracle Platform Security Services (OPSS), Oracle Fusion Middleware Audit Framework, and MDS, and services such as SSL and logging, supply custom WLST commands.

To use these custom WLST commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide* for more information.

4.5.1.1 Using WLST Commands for System Components

In addition to the commands provided by WLST for Oracle WebLogic Server, WLST provides a subset of commands to monitor and manage system components. These commands are:

- `startproc`(*componentName* [, *componentType*] [, *componentSet*): Starts the specified component.
- `stopproc`(*componentName* [, *componentType*] [, *componentSet*): Stops the specified component.
- `status`(*componentName* [, *componentType*] [, *componentSet*): Obtains the status of the specified component.
- `proclist()`: Obtains the list of components.
- `dumpMetrics`([*servers*,] [*format*]): Displays available metrics in the internal format or in XML.
- `displayMetricTables`([*metricTable_1*], [*metricTable_2*], [...,] [*servers*] [*variables*]): Displays the content of the DMS metric tables.
- `displayMetricTableNames`([*servers*]): Displays the names of the available DMS metric tables. The returned value is a string array containing metric table names.

Note: The `dmstool` command has been replaced with the following commands: `dumpMetrics`, `displayMetricTables`, `displayMetricTableNames`. For more information on DMS WLST commands, see "DMS Custom WLST Commands" in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*

4.6 DMS Spy Servlet

The DMS Spy servlet provides you with access to DMS metric data from a web browser. Data that is created and updated by DMS-enabled applications and components is accessible through the DMS Spy Servlet.

4.6.1 Viewing Performance Metrics Using the Spy Servlet

The DMS Spy Servlet is part of the DMS web application. The DMS web application's web archive file is dms.war, and can be found in the same directory as `dms.jar`: `<ORACLE_HOME>/modules/oracle.dms_11.1.1/dms.war`.

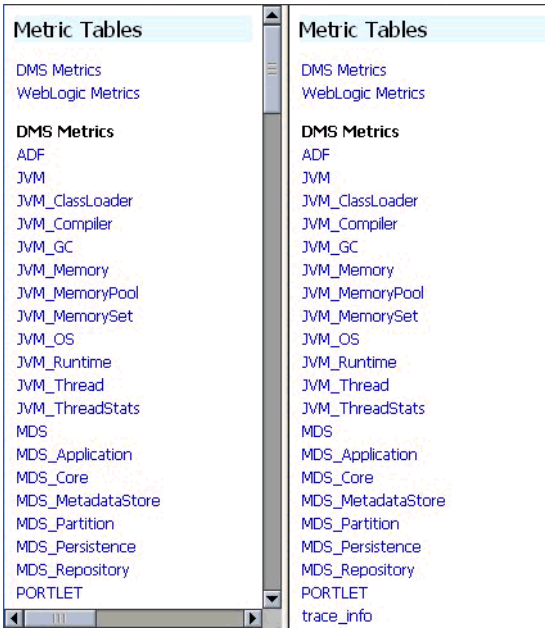
The DMS web application is deployed by default as part of a JRF-enabled server instance. The URL is: `http://host:port/dms/Spy`.

Only users who have Administrator role access can view this URL as access is controlled by standard J2EE elements in `web.xml`.

4.6.2 Using the DMS Spy Servlet

Figure 4–2 shows the initial page of the Spy servlet: both sides show the same list of metric tables.

Figure 4–2 Spy Servlet Page - Metrics Tables

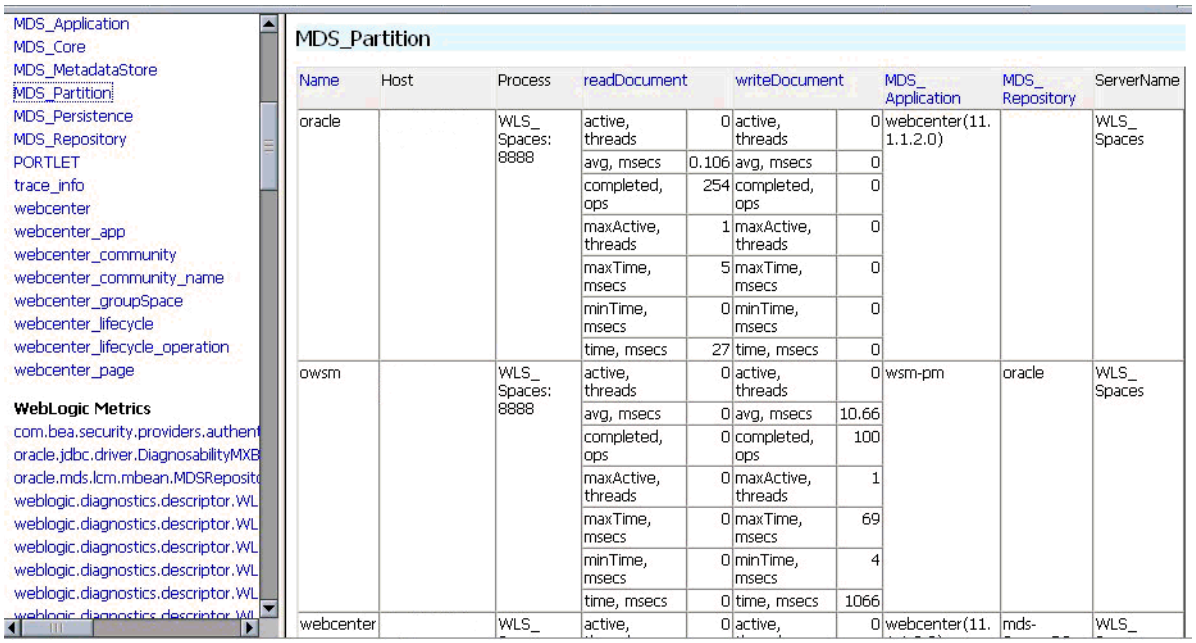


Note that the Spy servlet can display metric tables for WebLogic Server and also for non-J2EE components that are deployed.

For metric tables to appear in the Spy servlet, the component that creates and updates that table must be installed and running. Metric tables for components that are not running are not displayed. Metric tables with ":" in their name (for example, weblogic_j2eeserver:app_overview) are aggregated metric tables generated by metric rules.

To view the contents of a metric table, click the table name. For example, Figure 4–3 shows the MDS_Partition table.

Figure 4–3 MDS Partition Table



To get a description of the fields in a metric table, click the Metric Definitions link below the table.

4.7 Oracle Process Manager and Notification Server

Oracle Process Manager and Notification Server (OPMN) monitors the status of Oracle Fusion Middleware components. You can also start and stop system components, monitor system components, and perform many other tasks related to process management. For example, you can use OPMN to start and stop OPMN-managed processes, such as Oracle HTTP Server and Oracle Web Cache. For more information on OPMN commands, see "Section 5.5.4, "Monitoring Oracle HTTP Server".

Note: For more information on using OPMN, refer to *Oracle Fusion Middleware Oracle Process Manager and Notification Server Administrator's Guide*.

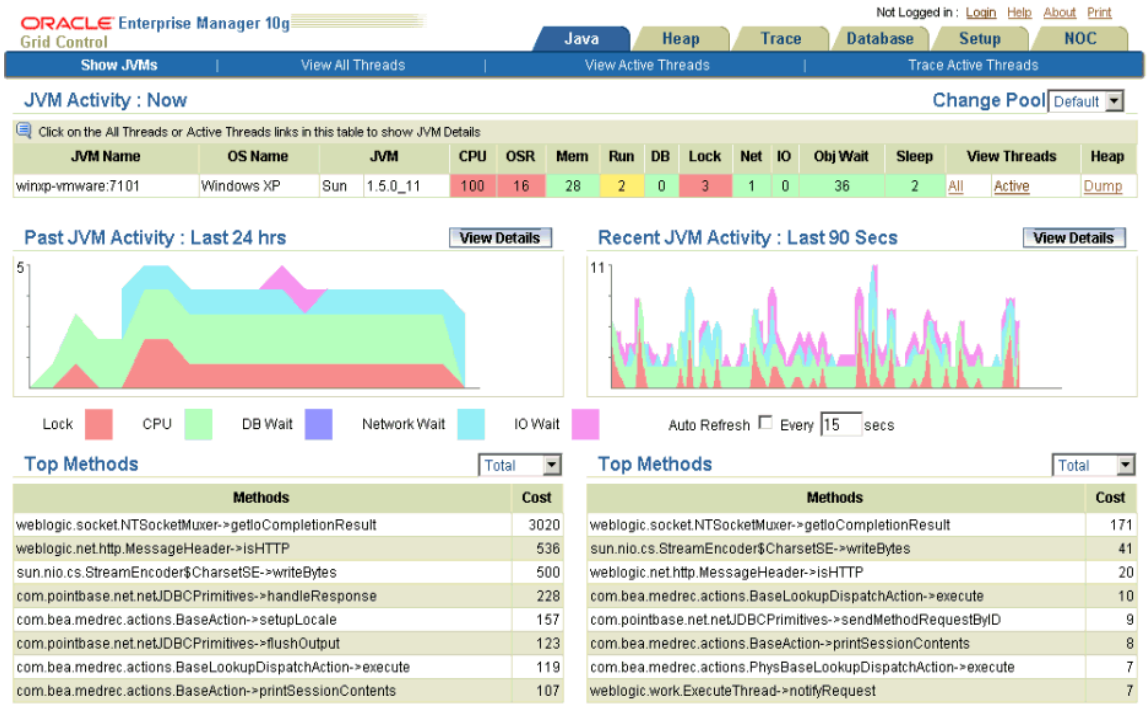
4.8 Oracle Enterprise Manager 11g Grid Control

While Fusion Middleware Control provides real-time performance monitoring for a single Fusion Middleware Farm, Oracle Enterprise Manager 11g Grid Control enables you to centrally manage multiple farms, in addition to the rest of your data center (such as the underlying host and operating system, databases, packaged applications such as Oracle E-Business Suite and Siebel, and third party products such as F5 BIG_IP Local Traffic Manager).

Note: Grid Control is not provided out-of-box with the Fusion Middleware installation; it requires a separate installation. Refer to the Oracle Enterprise Manager 11g Grid Control Installation and Advanced Configuration Guide for further details on how to install Grid Control.

Key features available from Grid Control that are applicable to and relevant in monitoring Fusion Middleware performance include the following:

- Monitor multiple WebLogic Server Domains from a single console.
- Out-of-the-box availability and performance monitoring
- Monitor availability and performance in real-time as well as from an historical perspective
- Specify warning versus critical thresholds for key performance metrics
- Receive email and/or page notifications when metric thresholds are reached
- Perform trend analysis on collected performance information
- Application Diagnostics for Java (AD4J) which provides production diagnostics with no application instrumentation. The AD4J screen is shown below.

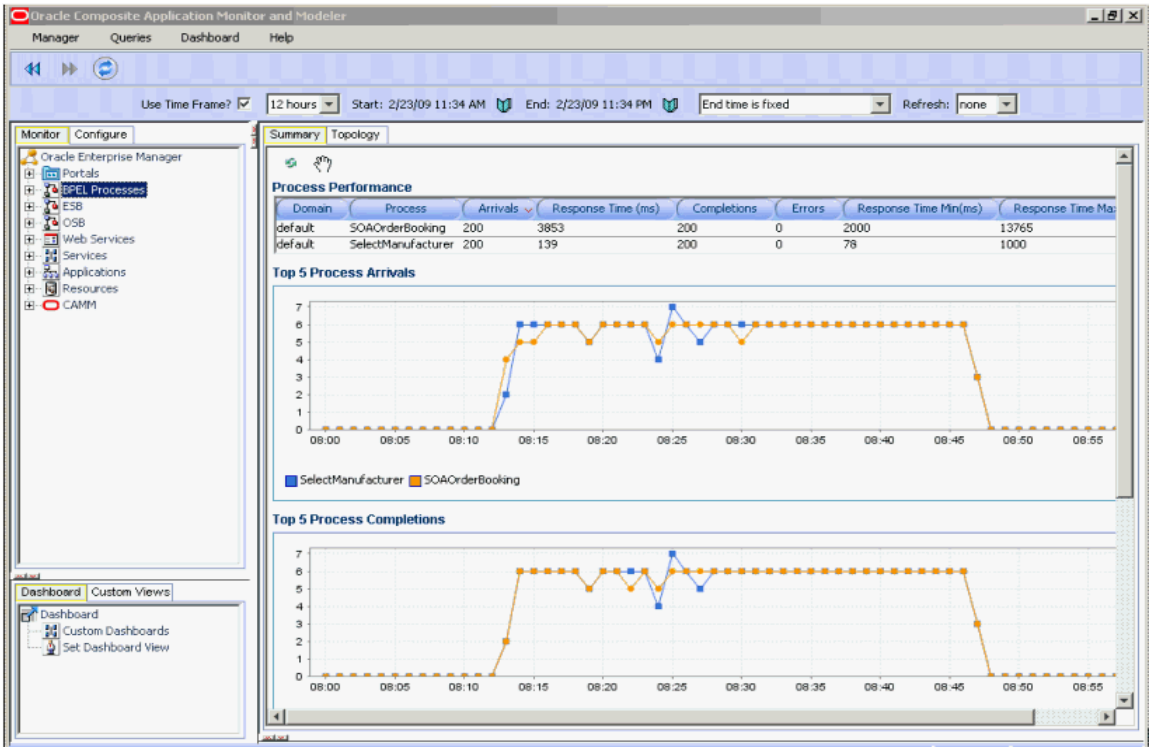


Key features of AD4J include the following:

- Full method, stack and thread state visibility
- Quick ranking of high-cost code being executed for bottleneck identification
- Line-of-code granularity
- Cross-tier database and EJB/RMI correlation
- Java thread lock and synchronization detection

- Thread activity tracing
- Heap snapshot and analysis
- Differential heap analysis to quickly isolate memory leaks
- Threshold based alerting
- Alert actions through SNMP traps, SMTP, or HTTP request

- Composite Application Monitor and Modeler (CAMM) provides application service management for complex composite services such as Portal, BPEL, ESB, OSB, and Web Services. The CAMM screen is shown below.



Key features of CAMM include the following:

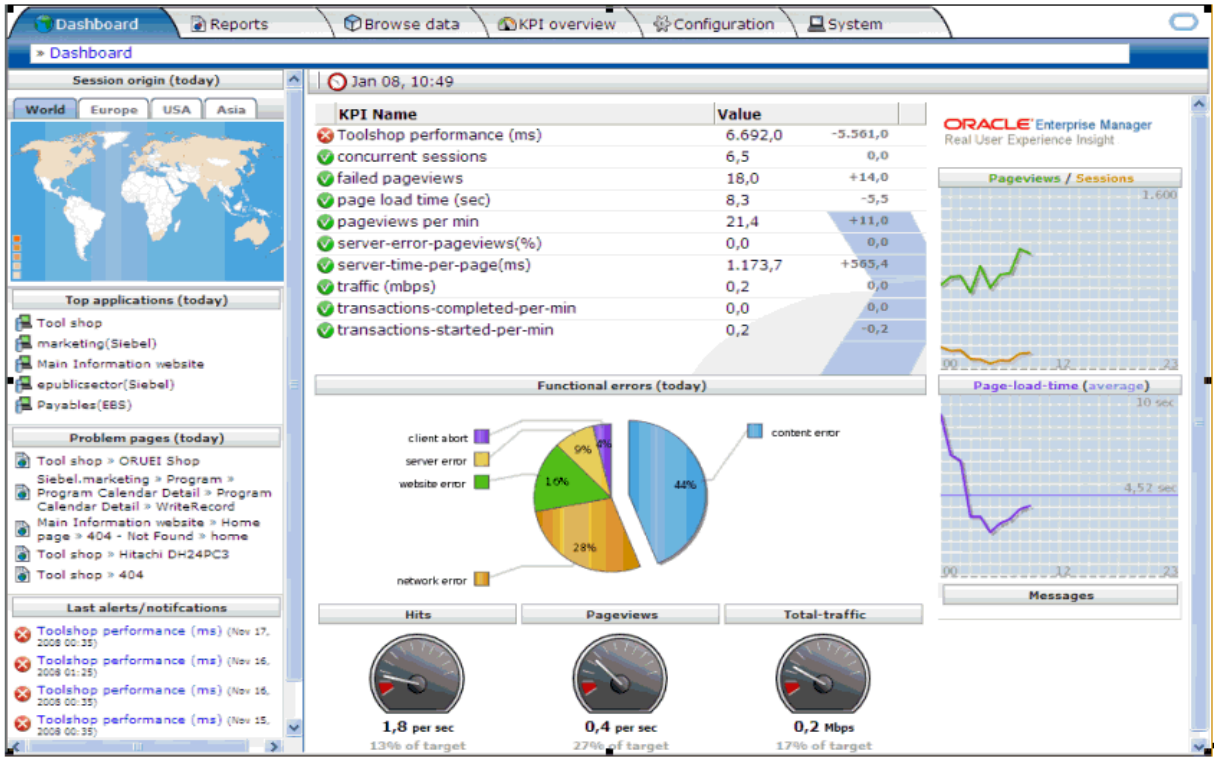
- Automated discovery of complex services (Portal, BPEL, ESB, OSB, Web Services)
- Automatic metadata analysis and monitoring instrumentation configuration
- run time dependency analysis between SOA services and endpoints
- Metadata model presents monitored targets using native terminology
- Invocation metrics based on both arrival and completion of inbound request
- Response time mean, max, and min metrics
- Tiered data aggregation for long term storage and trending
- Historical views into arbitrary date/time ranges
- Comparative views to compare any two arbitrary date/time ranges with simultaneous scrolling
- Custom views to combine arbitrary graphs, tables, and functional views
- Customizable hierarchy of custom views

  - Threshold based alerting on any metric, both individual and aggregate
  - Alert actions through SNMP traps, SMTP, logging, or custom scripts

- Real User Experience Insight (RUEI) provides insight into the real end-user experience for your applications. No application instrumentation is required. The RUEI screen is shown below.

  Key features of RUEI include the following:

  - Replay of poor performance
  - Executive dashboards
  - Extensive KPI and SLA monitoring
  - Full alerting capabilities
  - Quick bottle-neck analyses
  - Transaction performance analyses
  - Customized reporting
  - Trend analyses
  - Full data integration through XML



Note: For more information on the monitoring features available in Oracle Enterprise Manager 10g Grid Control, refer to the Oracle Enterprise Manager 10g Grid Control documentation available on the Oracle Technology Network here: http://www.oracle.com/technology/documentation/oem.html.

4.9 Native Operating System Performance Commands

Each operating system has native tools and utilities that can be useful for monitoring purposes. Native operating system commands enable you to gather and monitor for example CPU utilization, paging activity, swapping, and other system activity information.

For details on operating system commands, refer to the documentation provided by the operating system vendor.

4.10 Network Performance Monitoring Tools

Your operating system's network monitoring tools can be used to monitor utilization, verify that the network is not becoming a bottleneck, or detect packet loss or other network performance issues. For details on network performance monitoring, refer to your operating system documentation.

Part II

Core Components

This part describes configuring core components to improve performance. It contains the following chapters:

- Chapter 5, "Oracle HTTP Server Performance Tuning"
- Chapter 6, "Oracle Metadata Service (MDS) Performance Tuning"

Note: For information on performance tuning the Oracle WebLogic Server, see *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

5

Oracle HTTP Server Performance Tuning

This chapter discusses the techniques for optimizing Oracle HTTP Server performance. This chapter contains the following sections:

- Section 5.1, "About Oracle HTTP Server"
- Section 5.2, "Oracle HTTP Server Directives Tuning Considerations"
- Section 5.3, "Oracle HTTP Server Logging Options"
- Section 5.4, "Oracle HTTP Server Security Performance Considerations"
- Section 5.5, "Oracle HTTP Server Performance Tips"

> **Note:** The configuration examples and recommended settings described in this chapter are for illustrative purposes only. Consult your own use case scenarios to determine which configuration options can provide performance improvements.

5.1 About Oracle HTTP Server

Oracle HTTP Server (OHS) is the Web server component for Oracle Fusion Middleware. It provides a listener for Oracle WebLogic Server and the framework for hosting static pages, dynamic pages, and applications over the Web. Oracle HTTP Server is based on the Apache 2.2.x infrastructure, and includes modules developed specifically by Oracle. The features of single sign-on, clustered deployment, and high availability enhance the operation of the Oracle HTTP Server.

For more information see *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

For more information on the Apache open-source software infrastructure, see the Apache Software Foundation web site at `http://www.apache.org/`.

5.2 Oracle HTTP Server Directives Tuning Considerations

Oracle HTTP Server uses directives in `httpd.conf`. This configuration file specifies the maximum number of HTTP requests that can be processed simultaneously, logging details, and certain limits and time outs.

More information on configuring the Oracle HTTP Server, see "Management Tools for Oracle HTTP Server" in *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

Oracle HTTP Server supports three different Multi-Processing Modules (MPMs) by default. The MPMs supported are:

- Worker - This uses Multi-Process-Multi-Threads model and is the default MPM on all platforms other than Microsoft Windows platforms. Multi-thread support makes it more scalable by using fewer system resources and multi-process support makes it more stable.
- WinNT - This MPM is for Windows platforms only. It consists of a parent process and a child process. The parent process is the control process, and the child process creates threads to handle requests.
- Prefork - This is Apache 1.3.x style and uses processes instead of threads. This is considered the least efficient MPM.

The directives for each MPM type are defined in the `ORACLE_INSTANCE/config/OHSComponent/<ohsname>/httpd.conf file`. The default MPM type is Worker MPM. To use a different MPM (such as Prefork MPM), edit the `ORACLE_HOME/ohs/bin/apachectl` file.

Note: The information in this chapter is based on the use of Worker and WinNT MPMs, which use threads. The directives listed below may not be applicable if you are using the prefork MPM. If you are using Oracle HTTP Server based on Apache 1.3.x or Apache 2.2 with prefork MPM, refer to the Oracle Application Server 10*g* Release 3 documentation at `http://www.oracle.com/technology/documentation/appserver10132.html`.

Table 5–1 Oracle HTTP Server ***Configuration Properties***

Directive	Description
`ListenBackLog` This directive maps to the **Maximum Queue Length** field on the Performance Directives screen.	Specifies the maximum length of the queue of pending connections. Generally no tuning is needed. Note that some operating systems do not use exactly what is specified as the backlog, but use a number based on, but normally larger than, what is set. Default Value: 511
`MaxClients` This directive maps to the **Maximum Requests** field on the Performance Directives screen. Note that this parameter is not available in mod_winnt (Microsoft Windows). Winnt uses a single process, multi-threaded model and is controlled by `ThreadLimit` directive.	Specifies a limit on the total number of servers running, that is, a limit on the number of clients who can simultaneously connect. If the number of client connections reaches this limit, then subsequent requests are queued in the TCP/IP system up to the limit specified with the `ListenBackLog` directive (after the queue of pending connections is full, new requests generate connection errors until a thread becomes available). You can configure the `MaxClients` directive in the httpd.conf file up to a maximum of 8K (the default value is 150). If your system is not resource-saturated and you have a user population of more than 150 concurrent HTTP/Thread connections, you can improve your performance by increasing `MaxClients` to increase server concurrency. Increase MaxClients until your system becomes fully utilized (85% is a good threshold). Conversely, when system resources are saturated, increasing `MaxClients` does not improve performance. In this case, the `MaxClients` value could be reduced as a throttle on the number of concurrent requests on the server. If the server handles persistent connections, then it may require sufficient concurrent httpd or thread server processes to handle both active and idle connections. When you specify `MaxClients` to act as a throttle for system concurrency, you must consider that persistent idle httpd connections also consume httpd/thread processes. Specifically, the number of connections includes the currently active persistent and non-persistent connections and the idle persistent connections. A persistent, `KeepAlive`, http connection consumes an httpd child process, or thread, for the duration of the connection, even if no requests are currently being processed for the connection. If you have sufficient capacity, `KeepAlive` should be enabled; using persistent connections improves performance and prevents wasting CPU resources reestablishing HTTP connections. Normally, you should not change `KeepAlive` parameters. The maximum allowed value for `MaxClients` is 8192 (8K). Default Value: 150
`StartServers` This directive maps to the **Initial Child Server Processes** field on the Performance Directives screen.	Specifies the number of child server processes created on startup. If you expect a sudden load after restart, set this value based on the number child servers required. Note that the following parameters are inter-related and applicable only on UNIX platforms (worker_mpm): ■ `MaxClients` ■ `MaxSpareThreads` and `MinSpareThreads` ■ `ServerLimit` and `StartServers` On the Windows platform (mpm_winnt), as well as UNIX platforms, the following parameters are important to tune: ■ `ThreadLimit` ■ `ThreadsPerChild` Note that each child process has a set of child threads defined for them and that can actually handle the requests. Use `ThreadsPerChild` in connection with this directive. The values of `ThreadLimit`, `ServerLimit`, and `MaxClients` can indirectly affect this value. Read the notes for these directives and use them in conjunction with this directive. Default Value: 2

Table 5–1 (Cont.) Oracle HTTP Server ***Configuration Properties***

Directive	Description
`ServerLimit` Note that this parameter is not available in mod_winnt (Microsoft Windows). Winnt uses a single process, multi-threaded model	Specifies an upper limit on the number of server (child) processes that can exist or be created. This value overrides the `StartServers` value if that value is greater than the `ServerLimit` value. This is used to control the maximum number of server processes that can be created. Default Value: 16
`ThreadLimit`	Specifies the upper limit on the number of threads that can be created under a server (child) process. This value overrides the `ThreadsPerChild` value if that value is greater than the `ThreadLimit` value. This is used to control the maximum number of threads created per process to avoid conflicts/issues. Default Value: 64
`ThreadsPerChild` This directive maps to the **Threads Per Child Server Process** field on the Performance Directives screen.	Sets the number of threads created by each server (child) process at startup. Default Value: 64 when mpm_winnt is used and 25 when Worker MPM is used. The `ThreadsPerChild` directive works with other directives, as follows: At startup, Oracle HTTP Server creates a parent process, which creates several child (server) processes as defined by the `StartServers` directive. Each server process creates several threads (server/worker), as specified in `ThreadsPerChild`, and a listener thread which listens for requests and transfers the control to the worker/server threads. After startup, based on load conditions, the number of server processes and server threads (children of server processes) in the system are controlled by `MinSpareThreads` (minimum number of idle threads in the system) and `MaxSpareThreads` (maximum number of idle threads in the system). If the number of idle threads in the system is more than `MaxSpareThreads`, Oracle HTTP Server terminates the threads and processes if there are no child threads for a process. If the number of idle threads is fewer than `MinSpareThreads`, it creates new threads and processes if the `ThreadsPerChild` value has already been reached in the running processes. The following directives control the limit on the above directives. Note that the directives below should be defined before the directives above for them to take effect. ■ `ServerLimit` - Defines the upper limit on the number of servers that can be created. This affects `MaxClients` and `StartServers`. ■ `ThreadLimit` - Defines the upper limit on `ThreadsPerChild`. If `ThreadsPerChild` is greater than `ThreadLimit`, then it is automatically trimmed to the latter value. ■ `MaxClients` - Defines the upper limit on the number of server threads that can process requests simultaneously. This should be equal to the number of simultaneous connections that can be made. This value should be a multiple of `ThreadsPerChild`. If `MaxClients` is greater than `ServerLimit` multiplied by `ThreadsPerChild`, it is automatically be trimmed to the latter value.
`MaxRequestsPerChild` This directive maps to the **Max Requests Per Child Server Process** field on the Performance Directives screen.	Specifies the number of requests each child process is allowed to process before the child process dies. The child process ends to avoid problems after prolonged use when Apache (and any other libraries it uses) leak memory or other resources. On most systems, this is not needed, but some UNIX systems have notable leaks in the libraries. For these platforms, set `MaxRequestsPerChild` to 10000; a setting of 0 means unlimited requests. This value does not include `KeepAlive` requests after the initial request per connection. For example, if a child process handles an initial request and 10 subsequent "keep alive" requests, it would only count as 1 request toward this limit. Default Value: 0 **Note**: On Windows systems `MaxRequestsPerChild` should always be set to `0` (unlimited) since there is only one server process.

Table 5–1 (Cont.) Oracle HTTP Server ***Configuration Properties***

Directive	Description

`MaxSpareThreads`
`MinSpareThreads`
These directives map to the **Maximum Idle Threads** and **Minimum Idle Threads** fields on the Performance Directives screen.
Note that these parameters are not available in mod_winnt (Windows platform).
	Controls the server-pool size. Rather than estimating how many server threads you need, Oracle HTTP Server dynamically adapts to the actual load. The server tries to maintain enough server threads to handle the current load, plus a few additional server threads to handle transient load increases such as multiple simultaneous requests from a single browser.
	The server does this by periodically checking how many server threads are waiting for a request. If there are fewer than `MinSpareThreads`, it creates a new spare. If there are more than `MaxSpareThreads`, some of the spares are removed.
	Default Values:
	MaxSpareThreads: 75
	MinSpareThreads: 25

`Timeout` This directive maps to the **Request Timeout** field on the Performance Directives screen.	The number of seconds before incoming receives and outgoing sends time out. Default Value: 300
`KeepAlive` This directive maps to the **Multiple Requests Per Connection** field on the Performance Directives screen.	Whether or not to allow persistent connections (more than one request per connection). Set to Off to deactivate. Default Value: On
`MaxKeepAliveRequests`	The maximum number of requests to allow during a persistent connection. Set to 0 to allow an unlimited amount. If you have long client sessions, consider increasing this value. Default Value: 100
`KeepAliveTimeout` This directive maps to the **Allow With Connection Timeout (seconds)** field, which is located under the **Multiple Requests Per Connection** field, on the Performance Directives screen.	Number of seconds to wait for the next request from the same client on the same connection. Default Value: 15 seconds

5.2.1 How Persistent Connections Can Reduce Httpd Process Availability

If your browser supports persistent connections, you can support them on the server using the `KeepAlive` directives in the Oracle HTTP Server. Persistent Connections can improve performance by reducing the work load on the server. With Persistent Connections enabled, the server does not have to repeat the work to set up the connections with a client.

The default settings for the `KeepAlive` directives are:

```
KeepAlive on
MaxKeepAliveRequests 100
KeepAliveTimeOut 15
```

These settings allow enough requests per connection and time between requests to reap the benefits of the persistent connections, while minimizing the drawbacks. You should consider the size and behavior of your own user population when setting these

values. For example, if you have a large user population and the users make small infrequent requests, you may want to reduce the `keepAlive` directive default settings, or even set `KeepAlive` to off. If you have a small population of users that return to your site frequently, you may want to increase the settings.

`KeepAlive` option should be used judiciously along with `MaxClients` directive. `KeepAlive` option would tie a worker thread to an established connection until it times out or the number of requests reaches the limit specified by `MaxKeepAliveRequests`. This means that the connections or users in the `ListenBacklog` queue would be starving for a worker until the worker is relinquished by the keep-alive user. The starvation for resources happens on the `KeepAlive` user load with user population consistently higher than that specified in the `MaxClients`.

Note: The `Maxclients` property is applicable only to UNIX platforms. On Windows, the same functionality is achieved through the `ThreadLimit` and `ThreadsPerChild` parameters.

Increasing `MaxClients` may impact performance in the following ways:

- A high number of `MaxClients` can overload the system resources and may lead to poor performance.
- For a high user population with fewer requests, consider increasing the `MaxClients` to support `KeepAlive` connections to avoid starvation. Note that this can impact overall performance if the user concurrency increases. System performance is impacted by increased concurrency and can possibly cause the system to fail.

`MaxClients` should always be set to a value where the system would be stable or performing optimally (~85% CPU).

Typically for high user population with less frequent requests, consider turning the `KeepAlive` option off or reduce it to a very low value to avoid starvation.

Disabling the `KeepAlive` connection may impact performance in the following ways:

- Connection establishment for every request has a cost.
- If the frequency of creating and closing connections is higher, then some system resources are used. The TCP connection has a `time_wait` interval before it can close the socket connection and open file descriptors for every connection. The default `time_wait` value is 60 seconds and each connection can take 60 seconds to close, even after it is relinquished by the server.

WARNING: To avoid potential performance issues, values for any parameters should be set only after considering the nature of the workload and the system capacity.

5.3 Oracle HTTP Server Logging Options

This section discusses types of logging, log levels, and the performance implications for using logging.

5.3.1 Access Logging

Access logs are generally enabled to track who accessed what. The access_log file, available in the ORACLE_INSTANCE/diagnostics/logs/OHS/*ohsname* directory, contains an entry for each request that is processed. This file grows as time passes and can consume disk space. Depending on the nature of the workload, the access_log has little impact on performance. If you notice that performance is becoming an issue, the file can be disabled if some other proxy or load balancer is used and gives the same information.

5.3.2 Configuring the HostNameLookups Directive

By default, the HostNameLookups directive is set to Off. The server writes the IP addresses of incoming requests to the log files. When HostNameLookups is set to On, the server queries the DNS system on the Internet to find the host name associated with the IP address of each request, then writes the host names to the log. Depending on the server load and the network connectivity to your DNS server, the performance impact of the DNS HostNameLookup may be high. When possible, consider logging only IP addresses. On UNIX systems, you can resolve IP addresses to host names off-line, with the `logresolve` utility found in the `ORACLE_HOME/Apache/Apache/bin/` directory.

5.3.3 Error logging

The server notes unusual activity in an error log. The *ohsname*.log file, available in ORACLE_INSTANCE/diagnostics/logs/OHS/*ohsname* directory, contains errors, warnings, system information, and notifications (depending on the log-level setting).

The httpd.conf file contains the error log configuration for OHS. The logging mode is defined by the "OraLogMode" directive. The default is "odl-text", which produces the Oracle diagnostic logging format in a text file. Alternatively, change this to "odl-xml" to produce the Oracle diagnostic logging format in an XML file.

For Oracle diagnostic-style logging, "OraLogSeverity" directive is used for setting the log level.

For Apache-style logging, the ErrorLog and LogLevel directives identify the log file and the level of detail of the messages recorded. The default debug level is Warn.

Excessive logging can have some performance cost and may also fill disk space. The log level control should be used based on need. For requests that use dynamic resources, for example, requests that use `mod_osso` or `mod_plsql`, there is a performance cost associated with setting higher debugging levels, such as the debug level.

5.4 Oracle HTTP Server Security Performance Considerations

This section covers the following topics:

- Oracle HTTP Server Secure Sockets Layer (SSL) Performance Issues
- Oracle HTTP Server Port Tunneling Performance Issues

5.4.1 Oracle HTTP Server Secure Sockets Layer (SSL) Performance Issues

Secure Sockets Layer (SSL) is a protocol developed by Netscape Communications Corporation that provides authentication and encrypted communication over the Internet. Conceptually, SSL resides between the application layer and the transport

layer on the protocol stack. While SSL is technically an application-independent protocol, it has become a standard for providing security over HTTP, and all major web browsers support SSL.

SSL can become a bottleneck in both the responsiveness and the scalability of a web-based application. Where SSL is required, the performance challenges of the protocol should be carefully considered. Session management, in particular session creation and initialization, is generally the most costly part of using the SSL protocol, in terms of performance.

This section covers the following SSL performance-related information:

- Section 5.4.1.1, "Oracle HTTP Server SSL Caching"
- Section 5.4.1.2, "SSL Application Level Data Encryption"
- Section 5.4.1.3, "SSL Performance Recommendations"

See Also: *Oracle Fusion Middleware Security Guide*

5.4.1.1 Oracle HTTP Server SSL Caching

When an SSL connection is initialized, a session-based handshake between client and server occurs that involves the negotiation of a cipher suite, the exchange of a private key for data encryption, and server and, optionally, client, authentication through digitally-signed certificates.

After the SSL session state has been initiated between a client and a server, the server can avoid the session creation handshake in subsequent SSL requests by saving and reusing the session state. The Oracle HTTP Server caches a client's SSL session information by default. With session caching, only the first connection to the server incurs high latency.

The `SSLSessionCacheTimeout` directive in `ssl.conf` determines how long the server keeps a saved SSL session (the default is 300 seconds). Session state is discarded if it is not used after the specified time period, and any subsequent SSL request must establish a new SSL session and begin the handshake again. The `SSLSessionCache` directive specifies the location for saved SSL session information (the default location is the following directory):

```
$ORACLE_INSTANCE/diagnostics/logs/$COMPONENT_ TYPE/$COMPONENT_
NAME
```

Note that multiple Oracle HTTP Server processes can use a saved session cache file.

Saving SSL session state can significantly improve performance for applications using SSL. For example, in a simple test to connect and disconnect to an SSL-enabled server, the elapsed time for 5 connections was 11.4 seconds without SSL session caching. With SSL session caching enabled, the elapsed time for 5 round trips was 1.9 seconds.

The reuse of saved SSL session state has some performance costs. When SSL session state is stored to disk, reuse of the saved state normally requires locating and retrieving the relevant state from disk. This cost can be reduced when using HTTP persistent connections. Oracle HTTP Server uses persistent HTTP connections by default, assuming they are supported on the client side. In HTTP over SSL as implemented by Oracle HTTP Server, SSL session state is kept in memory while the associated HTTP connection is persisted, a process which essentially eliminates the performance impacts associated with SSL session reuse (conceptually, the SSL connection is kept open along with the HTTP connection). For more information see Section 5.2.1, "How Persistent Connections Can Reduce Httpd Process Availability".

5.4.1.2 SSL Application Level Data Encryption

In most applications using SSL, the data encryption cost is small compared with the cost of SSL session management. Encryption costs can be significant where the volume of encrypted data is large, and in such cases the data encryption algorithm and key size chosen for an SSL session can be significant. In general there is a trade-off between security level and performance.

Oracle HTTP Server negotiates a cipher suite with a client based on the SSLCipherSuite attribute specified in ssl.conf. OHS 11*g* uses 128 bit Encryption algorithm by default and no longer supports lower encryption. Note that the previous release [10.1.3x] used 64 bit encryption for Windows. For UNIX, the 10.x releases had 128 bit encryption used by default.

> **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server* for information on using supported cipher suites.

5.4.1.3 SSL Performance Recommendations

The following recommendations can assist you with determining performance requirements when working with Oracle HTTP Server and SSL.

1. The SSL handshake is an inherently resource intensive process in terms of both CPU usage and response time. Thus, use SSL only where needed. Determine the parts of the application that require the security, and the level of security required, and protect only those parts at the requisite security level. Attempt to minimize the need for the SSL handshake by using SSL sparingly, and by reusing session state as much as possible. For example, if a page contains a small amount of sensitive data and several non-sensitive graphic images, use SSL to transfer the sensitive data only, use normal HTTP to transfer the images. If the application requires server authentication only, do not use client authentication. If the performance goals of an application cannot be met by this method alone, additional hardware may be required.
2. Design the application to use SSL efficiently. Group secure operations to take advantage of SSL session reuse and SSL connection reuse.
3. Use persistent connections, if possible, to minimize cost of SSL session reuse.
4. Tune the session cache timeout value (the `SSLSessionCacheTimeout` directive in `ssl.conf`). A trade-off exists between the cost of maintaining an SSL session cache and the cost of establishing a new SSL session. As a rule, any secured business process, or conceptual grouping of SSL exchanges, should be completed without incurring session creation more than once. The default value for the `SSLSessionCacheTimeout` attribute is 300 seconds. It is a good idea to test an application's usability to help tune this setting.
5. If large volumes of data are being protected through SSL, pay close attention to the cipher suite being used. The `SSLCipherSuite` directive specified in `ssl.conf` controls the cipher suite. If lower levels of security are acceptable, use a less-secure protocol using a smaller key size (this may improve performance significantly). Finally, test the application using each available cipher suite for the specified security level to find the optimal suite.
6. If SSL remains a bottleneck to the performance and scalability of your application, after taking the preceding considerations into account, consider deploying multiple Oracle HTTP Server instances over a hardware cluster or consider the use of SSL accelerator cards.

5.4.2 Oracle HTTP Server Port Tunneling Performance Issues

When OracleAS Port Tunneling is configured, every request processed passes through the OracleAS Port Tunneling infrastructure. Thus, using OracleAS Port Tunneling can have an impact on the overall Oracle HTTP Server request handling performance and scalability.

With the exception of the number of OracleAS Port Tunneling processes to run, the performance of OracleAS Port Tunneling is self-tuning. The only performance control available is to start more OracleAS Port Tunneling processes; this increases the number of available connections and the scalability of the system.

The number of OracleAS Port Tunneling processes is based on the degree of availability required, and the number of anticipated connections. This number cannot be automatically determined because for each additional process a new port must be opened through the firewall between the DMZ and the intranet. You cannot start more processes than you have open ports, and you do not want less processes than open ports, since in this case ports would not have any process bound to them.

To measure the OracleAS Port Tunneling performance, determine the request time for servlet requests that pass through the OracleAS Port Tunneling infrastructure. The response time running with OracleAS Port Tunneling should be compared with a system without OracleAS Port Tunneling to determine whether your performance requirements can be met using OracleAS Port Tunneling.

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server* for information on configuring OracleAS Port Tunneling

---

5.5 Oracle HTTP Server Performance Tips

The following tips can enable you to avoid or debug potential Oracle HTTP Server performance problems:

- Analyze Static Versus Dynamic Requests
- Beware of a Single Data Point Yielding Misleading Results
- Beware of Having More Modules
- Monitoring Oracle HTTP Server

5.5.1 Analyze Static Versus Dynamic Requests

It is important to understand where your server is spending resources so you can focus your tuning efforts in the areas where the most stands to be gained. In configuring your system, it can be useful to know what percentage of the incoming requests are static and what percentage are dynamic.

Generally, you want to concentrate your tuning effort on dynamic pages because dynamic pages can be costly to generate. Also, by monitoring and tuning your application, you may find that much of the dynamically generated content, such as catalog data, can be cached, sparing significant resource usage.

5.5.2 Beware of a Single Data Point Yielding Misleading Results

You can get unrepresentative results when data outliers appear. This can sometimes occur at start-up. To simulate a simple example, assume that you ran a PL/SQL "Hello, World" application for about 30 seconds. Examining the results, you can see that the work was all done in `mod_plsql.c`:

```
/ohs_server/ohs_module/mod_plsql.c
  handle.maxTime:     859330
  handle.minTime:      17099
  handle.avg:          19531
  handle.active:           0
  handle.time:      24023499
  handle.completed:     1230
```

Note that `handle.maxTime` is much higher than `handle.avg` for this module. This is probably because when the first request is received, a database connection must be opened. Later requests can make use of the established connection. In this case, to obtain a better estimate of the average service time for a PL/SQL module, that does not include the database connection open time which causes the `handle.maxTime` to be very large, recalculate the average as in the following:

```
(time - maxTime)/(completed -1)
```

For example:

```
(24023499 - 859330)/(1230 - 1) = 18847.98
```

5.5.3 Beware of Having More Modules

Oracle HTTP Server, which is now based on Apache 2.2, has a slight change in architecture in the way the requests are handled, compared to the previous release of Oracle HTTP Server, which was based on Apache 1.3.

In the new architecture, Oracle HTTP Server invokes the service function of each module that is loaded (in the order of definition in `httpd.conf` file) until the request is serviced. This indicates that there is some cost associated with invoking the service function of each module, to know if the service is accepted or declined.

Because of this change in architecture, consider placing the most frequently hit modules above the others in the `httpd.conf` file.

Finally, for the static page requests, which are directly deployed to Oracle HTTP Server and served by the default handler, the request has to go through all the modules before the default handler is invoked. This process can impact performance of the request so consider enabling only the modules that are required by the deployed application. Example, if "mod_plsql" is never used by the deployed application, disable it to maintain performance.

In addition, there are a few modules that register their hooks to do some work during the URL translation phase, which would add to the cost of request processing time. Example: mod_security, when enabled, has a cost of about 10% on CPU Cost per Transaction for the specweb benchmark. Again, you should enable only those modules that are required by your deployed applications to save CPU time.

5.5.4 Monitoring Oracle HTTP Server

Oracle Fusion Middleware automatically and continuously measures run-time performance for Oracle HTTP Server. The performance metrics are automatically enabled; you do not need to set options or perform any extra configuration to collect them. If you encounter a problem, such as an application that is running slowly or is hanging, you can view particular metrics to find out more information about the problem.

Note: Fusion Middleware Control provides real-time data. For more information on using Fusion Middleware Control to view performance metrics for HTTP Server, see "Monitoring Oracle HTTP Server Performance" in *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

If you are interested in viewing historical data, consider using Grid Control. See Section 4.8, "Oracle Enterprise Manager 11g Grid Control".

In addition to the Fusion Middleware Control, Oracle HTTP Server also has Dynamic Monitoring Service (DMS), which collects metrics for every functional piece. You can review these metrics as needed to understand system behavior at a given point of time. This displays memory, CPU information and the min, max, average times for the request processing at every layer in Oracle HTTP Server. The metrics also display details about load level, number of threads, number of active connections, and so on, which can help in tuning the system based on real usage.

You can use Oracle Enterprise Manager or SpyServlet to monitor the metrics. See Chapter 4, "Monitoring Oracle Fusion Middleware". Another way to view DMS metrics for OHS is shown in the following example:

1. `cd $INSTANCE_HOME/bin`
2. `./opmnctl metric op=query COMPONENT_NAME=<component_name> dmsarg=[name=/OHS/Modules/<module_name>.c`

   Examples:

   ```
   ./opmnctl metric op=query COMPONENT_NAME=ohs1
   dmsarg=[name=/OHS/Modules/mod_cgi.c
   ```

   ```
   ./opmnctl metric op=query COMPONENT_NAME=ohs1 dmsarg=[name=*]
   ```

6

Oracle Metadata Service (MDS) Performance Tuning

This chapter provides tuning tips for Oracle Metadata Service (MDS).

- Section 6.1, "About Oracle Metadata Services (MDS)"
- Section 6.2, "Tuning Database Repository"
- Section 6.3, "Purging Document Version History"
- Section 6.4, "Using Database Polling Interval for Change Detection"
- Section 6.5, "Tuning Cache Configuration"
- Section 6.6, "Analyzing Performance Impact from Customization"
- Section 6.7, "Understanding DMS metrics and Characteristics"

6.1 About Oracle Metadata Services (MDS)

Oracle Metadata Services (MDS) is an application server and Oracle relational database that keeps metadata in these areas: a file-based repository data, dictionary tables (accessed by built-in functions) and a metadata registry. One of the primary uses of MDS is to store customizations and persisted personalization for Oracle applications. Oracle Metadata Services (MDS) is used by components such as Oracle WebCenter Framework and Oracle Application Development Framework (ADF) to manage metadata. Examples of metadata objects managed by MDS are: JSP pages and page fragments, ADF page definitions and task flows, and customized variants of those objects.

Note: Most of the Oracle Metadata Service configuration parameters are immutable and cannot be changed at run time unless otherwise specified.

6.2 Tuning Database Repository

For optimal performance of MDS APIs, the database schema for the MDS repository must be monitored and tuned by the database administrator. This section lists some recommended actions to tune the database repository:

- Collect Schema Statistics
- Increase Redo Log Size
- Reclaim Disk Space

- Monitor the Database Performance

For additional information on tuning the database, see "Optimizing Instance Performance" in *Oracle Database Performance Tuning Guide*.

6.2.1 Collect Schema Statistics

While MDS provides database indexes, they may not be used as expected due to a lack of schema statistics. If performance is an issue with MDS operations such as accessing or updating metadata in database repository, the database administrator must ensure that the statistics are available and current.

The following example shows one way that the Oracle database schema statistics can be collected:

```
execute dbms_stats.gather_schema_stats(ownname => <username>,
estimate_ percent => dbms_stats.auto_sample_size,
method_opt=> 'for all columns size auto',
cascade=>true);
```

For additional information on gathering statistics, see 'Automatic Performance Statistics" in *Oracle Database Performance Tuning Guide*.

6.2.2 Increase Redo Log Size

The size of the redo log files can influence performance, because the behavior of the database writer and archiver processes depend on the redo log sizes. Generally, larger redo log files provide better performance. Undersized log files increase checkpoint activity and reduce performance.

For more information see "Sizing Redo Log Files" in *Oracle Database Performance Tuning Guide*.

6.2.3 Reclaim Disk Space

While manual and auto purge operations delete the metadata content from the repository, the database may not immediately reclaim the space held by tables and indexes. This may result in the disk space consumed by MDS schema growing. Database administrators can manually rebuild the indexes and shrink the tables to increase performance and to reclaim disk space.

For more information see "Reclaiming Unused Space" in *Oracle Database Performance Tuning Guide*.

6.2.4 Monitor the Database Performance

Database administrators must monitor the database (for example, by generating automatic workload repository (AWR) reports for Oracle database) to observe lock contention, I/O usage and take appropriate action to address the issues.

For more information see:

- "Generating Automatic Workload Repository Reports" in *Oracle Database Performance Tuning Guide*
- "Monitoring Performance" in *Oracle Database Administrator's Guide*.

6.3 Purging Document Version History

MDS keeps document version history in the database's metadata store. As version history accumulates, it requires more disk space and degrades read/write performance. Assuming the document versions are not part of an active label, there are two ways to purge version history:

- Auto Purge
- Manual Purge

> **Note:** Purging version history manually may impact performance depending on the number of metadata updates that have been made since the last purge.

6.3.1 Auto Purge

The auto-purge interval can be configured or changed post deployment through MBeans. This element maps to the `AutoPurgeTimeToLive` attribute of the `MDSAppConfig` MBean. If your application uses the database store for MDS, you can set auto-purge by adding this entry in adf-config.xml prior to packaging the EAR:

```
<persistence-config>
  <auto-purge seconds-to-live="T"/>
</persistence-config>
```

In the example above, the auto-purge interval removes versions that are older than the specified time *T* (in seconds). For more information, see "Changing MDS Configuration Attributes for Deployed Applications" in *Oracle Fusion Middleware Administrator's Guide*.

> **Tip:** Adjust the auto-purge interval based on document versions created in your application. Purging can take longer based on number of versions created. See also "Setting MDS Cache Size and Purge Rate" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

6.3.2 Manual Purge

When you suspect that the database is running out of space or performance is becoming slower, you can manually purge existing version history using `WLST` command or through Oracle Enterprise Manager. Manual purging may impact performance, so plan to purge in a maintenance window or when the system is not busy.

For more information about manually purging version history, see "Purging Metadata Version History" in *Oracle Fusion Middleware Administrator's Guide*.

6.4 Using Database Polling Interval for Change Detection

MDS employs a polling thread which queries the database to gauge if the data in the MDS in-memory cache is out of sync with data in the database. This can happen when metadata is updated in another JVM. If it is out of sync, MDS clears any out of date-cached data so subsequent operations see the latest versions of the metadata. MDS invalidates the document cache, as well as MDS cache, so subsequent operations have the latest version of the metadata.

The polling interval can be configured or changed post deployment through MBeans. The element maps to the `ExternalChangeDetection` and `ExternalChangeDetectionInterval` attributes of the `MDSAppConfig` MBean. Prior to packaging the Enterprise ARchive (EAR) file, you can configure the polling interval by adding this entry in adf-config.xml:

```
<mds-config>
  <persistence-config>
    <external-change-detection enabled="true" polling-interval-secs="T"/>
  </persistence-config>
</mds-config>
```

In the example above, 'T' specifies the polling interval in seconds. The minimum value is 1. Lower values cause metadata updates, that are made in other JVMs, to be seen more quickly. It is important to note, however, that a lower value can also create increased middle tier and database CPU consumption due to the frequent queries. By default, polling is enabled ('true') and the default value of 30 seconds should be suitable for most purposes. For more information, see "Changing MDS Configuration Attributes for Deployed Applications" in *Oracle Fusion Middleware Administrator's Guide* ".

Note: When setting the polling interval, consider the following: if you poll too frequently, the database is queried for out-of-date versions; too infrequently, and those versions may stack up and polling can take longer to process.

6.5 Tuning Cache Configuration

MDS uses a cache to store metadata objects and related objects (such as XML content) in memory. MDS Cache is a shared cache that is accessible to all users of the application (on the same JVM). If a metadata object is requested repeatedly, with the same customizations, that object may be retrieved more quickly from the cache (a "warm" read). If the metadata object is not found in the cache (a "cold" read), then MDS may cache that object to facilitate subsequent read operations depending on the cache configuration, the type of metadata object and the frequency of access.

Cache can be configured or changed post deployment through MBeans. This element maps to the `MaximumCacheSize` attribute of the `MDSAppConfig` mbean. For more information see "Changing MDS Configuration Attributes for Deployed Applications" in *Oracle Fusion Middleware Administrator's Guide*.

Note: MDS Metrics, visible in Enterprise Manager, are useful for tuning the MDS cache. In particular, "`IOs Per MO Content Get`" or "`IOs Per Metadata Object Get`" should be less than 1. If not, consider increasing the size of the MDS cache. For more information on viewing DMS metric information, see Section 6.7, "Understanding DMS metrics and Characteristics".

Having a correctly sized cache can significantly improve throughput for repeated reading of metadata objects. The optimal cache size depends on the number of metadata objects used and the individual sizes of these objects. Prior to packaging the Enterprise ARchive (EAR) file, you can manually update the cache-config in adf-config.xml, by adding the following entry:

```
<mds-config>
  <cache-config>
```

```
    <max-size-kb>200000</max-size-kb>
  </cache-config>
</mds-config>
```

> **Note:** MDS cache grows in size as metadata objects are accessed until it hits `max-size-kb`. After that, objects are removed from the cache to make room as needed on a least recently used (LRU) basis to make room for new objects. Unless time-to-live (TTL) is set, the MDS cache continues to occupy the `max-size-kb` of memory.

6.5.1 Document Cache

In addition to the main MDS cache, MDS uses a document cache in conjunction with each metadata store to store thumbnail information about metadata documents (base document and customization documents) in memory. The entry for each document is small (<100 bytes) and the cache size limit is specified in terms of the number of document entries. MDS calculates an appropriate default size limit for the document cache based on the configured maximum size of the MDS Cache, as follows:

- If MDS cache is disabled, MDS defaults to having no document cache.
- If MDS cache is enabled, MDS defaults the document cache size to one document entry per KB of document cache configured.
- If cache-config is not specified, MDS defaults to 10000 document entries.
- If MDS cache is set to a very small value, MDS uses a minimum size of 500 for document cache.

In general, the defaults should be sufficient in most cases. However, insufficient document cache size may impact performance. Prior to packaging the Enterprise ARchive (EAR) file, you can explicitly set document cache size by adding this entry to adf-config.xml:

```
<metadata-store-usage id="db1">
  <metadata-store …>
    <property name = …/>
  </metadata-store>
  <document-cache max-entries="10000"/>
</metadata-store-usage>
```

> **Note:** Document cache is cleared when it exceeds the `document-cache max-entries` value. To avoid performance issues, consider increasing the document cache size if you receive a notification like the following for example:
>
> ```
> NOTIFICATION: Document cache DBMetadataStore : MDS
> Repository connection = <> exceeds its maximum
> number of entries <NNNN>, so the cache is cleared.
> ```

The DMS metric "`IOs Per Document Get`" (visible in Enterprise Manager, see Section 6.7) should be less than 1. If not, consider increasing the document cache size.

6.6 Analyzing Performance Impact from Customization

MDS customization may impact performance at run time.The impact from customization depends on many factors including:

- The type of customization that has been created (shared or user level)
- The percentage of metadata objects in the system which is customized. The lower this percentage the lower the impact of customization.
- The number of configured customization layers, and the efficiency of the customization classes.

There are two main types of customization:

- Shared Customizations: these are layers of customization corresponding to customization classes whose `getCacheHint` method returns `ALL_USERS` or `MULTI_USER`, meaning the layer applies to all or multiple users. Shared customizations are cached in the (shared) MDS cache.
- User Level Customizations (also known as Personalizations): these are layers of customization corresponding to customization classes whose `getCacheHint` method returns `SINGLE_USER`, meaning the layer applies to just one user. User customizations are generally cached on the user's session (HttpSession) until the user logs out.

For more information about customization concepts, writing customization classes, and configuring customization classes, see "Customizing Applications with MDS" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

6.7 Understanding DMS metrics and Characteristics

MDS uses DMS sensors to provide tuning and diagnostic information which can be viewed using Enterprise Manager. This information is useful, for example, to see if the MDS caches are large enough.

Information on DMS metrics can be found in the Fusion Middleware Control Console. Click **Help** at the top of the page to get more information. In most cases, the Help window displays a help topic about the current page. Click **Contents** in the Help window to browse the list of help topics, or click **Search** to search for a particular word or phrase.

Part III

Oracle Fusion Middleware Server Components

This part describes configuring Oracle Fusion Middleware server components to improve performance. It contains the following chapters:

- Chapter 7, "Oracle Application Development Framework Performance Tuning"
- Chapter 8, "Oracle TopLink (EclipseLink) JPA Performance Tuning"
- Chapter 9, "Oracle Web Cache Performance Tuning"

7

Oracle Application Development Framework Performance Tuning

This chapter provides basic guidelines on how to maximize the performance and scalability of the Oracle Application Development Framework (ADF). This chapter covers design, configuration, and deployment performance considerations in the following sections:

- Section 7.1, "About Oracle ADF"
- Section 7.2, "Oracle ADF View Performance"
- Section 7.3, "ADF Server Performance"

This chapter assumes that you are familiar with building ADF applications. To learn about ADF, see the following guides:

- *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Java EE Developer's Guide for Oracle Application Development Framework*

7.1 About Oracle ADF

Oracle Application Development Framework (Oracle ADF) is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications. Oracle ADF is suitable for enterprise developers who want to create applications that search, display, create, modify, and validate data using web, wireless, desktop, or web services interfaces. If you develop enterprise solutions that search, display, create, modify, and validate data using web, wireless, desktop, or web services interfaces, Oracle ADF can simplify your job. Used in tandem, Oracle JDeveloper 11g and Oracle ADF give you an environment that covers the full development lifecycle from design to deployment, with drag-and-drop data binding, visual UI design, and team development features built-in.

For more information see "Introduction to Oracle ADF" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

7.2 Oracle ADF View Performance

Oracle ADF Faces Rich Client (RC) is a set of standard JSF components that includes Ajax (Asynchronous JavaScript and XML) functionality.

While Ajax enables rich client-like applications to run on standard internet technologies, JSF provides server-side control, which reduces the dependency on an abundance of JavaScript often found in typical Ajax applications. Using Apache MyFaces Trinidad as the foundation, Oracle ADF Faces RC adds Ajax functionality, bringing rich Internet application (RIA) capabilities to JSF applications.

Before building, configuring, and deploying ADF applications, review the following topics to achieve optimal performance:

- Oracle ADF Faces Configuration and Profiling
- Performance Considerations for ADF Faces
- Tuning ADF Faces Component Attributes
- Performance Considerations for Table and Tree Components
- Performance Considerations for autoSuggest
- Data Delivery - Lazy versus Immediate
- Performance Considerations for DVT Components

7.2.1 Oracle ADF Faces Configuration and Profiling

This section discusses the configuration and profiling concepts of the ADF Faces. Configuration options for Oracle ADF Faces are set in the `web.xml` file. Most of these have default values that are tuned for performance. Table 7–1 describes some of these configuration options.

Table 7–1 ADF Configuration Options

Parameter	Description
`org.apache.myfaces.trinidad.resource.DEBUG`	Controls whether output should be enhanced for debugging or not. This parameter should be removed or set to `False`.
`oracle.adf.view.rich.CHECK_FILE_MODIFICATION`	Controls whether ADF faces check for modification date of JSP pages and discard any saved state if the file is changed. This parameter should be removed or set to `False`.
`oracle.adf.view.rich.CLIENT_STATE_METHOD`	Specifies which type of saving (`all` or `token`) should be used when client-side state saving is enabled. The default value is `token`.
`oracle.adf.view.rich.LOGGER_LEVEL`	Sets the log level on the client side. The default value is `OFF`. This parameter should be removed or set to `False`.
`oracle.adf.view.rich.ASSERT_ENABLED`	Specifies whether to process assertions on the client side. The default value is `OFF`. This parameter should be removed or set to `False`.

Note: When you are profiling or measuring client response time using the Firefox browser, ensure that the Firebug plug-in is disabled. While this plug-in is very useful for getting information about the page and for debugging JavaScript code on the page, it can impact the total response time.

For more information on disabling the Firefox Firebug plug-in, see the Firefox Support Home Page at `http://support.mozilla.com/en-US/kb/`.

7.2.2 Performance Considerations for ADF Faces

Table 7–2 provides configuration recommendations that may improve performance of ADF Faces:

Table 7–2 Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Use partial page navigation.	Partial Page Navigation is a feature of the ADF Faces framework that enables navigating from one ADF Faces page to another without a full page transition in the browser.The new page is sent to the client using Partial Page Rendering (PPR)/Ajax channel. The main advantage of partial page navigation over traditional full page navigation is improved performance: the browser no longer re-interprets and re-executes Javascript libraries, and does not spend time for cleanup/initialization of the full page. The performance benefit from this optimization is very big; it should be enabled whenever possible. Some known limitations of this feature are: ■ For the document's "metaContainer" facet (the HEAD section), only scripts are brought over with the new page. Any other content, such as icon links or style rules can be ignored. ■ Applications cannot use anchor (hash) URLs for their own purposes.
Use page templates.	Page templates enable developers to build reusable, data-bound templates that can be used as a shell for any page. A developer can build one or more templates that provide structure and consistency for other developers building web pages. The templates have both static areas on them that cannot be changed when they are used and dynamic areas on them where the developer can place content specific to the page they are building. There are some important considerations when using templates: ■ Since templates are present in every application page, they have to be optimized so that common performance impacts are avoided. Adding round corners to the template, for example, can impact the performance for every page. ■ When building complex templates, sometimes it is easier to build them in multiple pieces and include them in the top-level template using `<f:subview>` tag. However, from a performance perspective, this is not typically recommended since it can impact memory usage on the server side. (`<f:subview>` introduces another level into the ID scoping hierarchy, which results in longer IDs. Long IDs have a negative impact on performance. Developers are advised to avoid using `<f:subview>` unless it is required. It is not necessary to use `<f:subview>` around `<jsp:include>` if you can ensure that all IDs are unique. For example, if you are using `<jsp:include>`, break a large page into multiple pieces for easier editing. And whenever possible, avoid using `<f:subview>`. If you are including content developed by someone else, use `<f:subview>` if you do not know which IDs the developer used. In addition, you do not have to put `<f:subview>` at the top of a region definition. ■ Avoid long IDs in all cases, especially on pageTemplates, subviews, subforms, and on tables or within tables. Long IDs can have a performance impact on the server side, network traffic, and client processing.

Table 7–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Enable ADF rich client geometry management.	ADF Rich Client supports geometry management of the browser layout where parent components are in the UI explicitly. The children components are sized to stretch and fill up available space in the browser. While this feature makes the UI look better, it has a cost. The impact is on the client side where the browser must spend time resizing the components. The components that have geometry management by default are: PanelAccordion PanelStretchLayout PanelTabbed BreadCrumbs NavigationPane PanelSplitter Toolbar Toolbox Table Train **Notes:** ■ When using geometry management, try minimizing the number of child components that are under a parent geometry managed component. ■ The cost of geometry management is directly related to the complexity of child components. ■ The performance cost of geometry management can be smaller (as perceived by the user) for the pages with table or other data stamped components when table data streaming is used. The client-side geometry management can be executed while the browser is waiting for the data response from the server.
Use the ADF rich client overflow feature.	ADF Rich Client supports overflow feature. This feature moves the child components to the non-visible overflow area if they cannot fit the page. The components that have built-in support for overflow are: PanelTabbed, BreadCrumbs, NavigationPane, PanelAccordion, Toolbar, and Train. Toolbar should be contained in a Toolbox to handle the overflow. While there were several optimizations done to reduce the cost of overflow, it is necessary to pay special attention to the number of child components and complexity of each of them in the overflow component. Sometimes it is a good practice to set a big enough initial size of the overflow component such that overflow does not happen in most cases.

Table 7–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Use ADF Rich Client Partial Page Rendering (PPR).	ADF Rich Client is based on Asynchronous JavaScript and XML (Ajax) development technique. Ajax is a web development technique for creating interactive web applications, where web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, without the whole web page being reloaded. The effect is to improve a web page's interactivity, speed, and usability. With ADF Faces, the feature that delivers the Ajax partial page refresh behavior is called partial page rendering (PPR). PPR enables small areas of a page to be refreshed without having to redraw the entire page. For example, an output component can display what a user has chosen or entered in an input component or a command link or button can cause another component on the page to be refreshed. Two main Ajax patterns are implemented with partial page rendering (PPR): ■ native component refresh ■ cross-component refresh While the framework builds in native component refresh, cross-component refresh has to be done by developers in certain cases. Cross-component refresh is implemented declaratively or programmatically by the application developer defining which components are to trigger a partial update and which other components are to act as partial listeners, and so be updated. Using cross-component refresh and implementing it correctly is one of the best ways to improve client-side response time. While designing the UI page always think about what should happen when the use clicks a command button. Is it needed for the whole page to be refreshed or just an output text field? What should happen if the value in some field is updated? For more information, refer to *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*). Consider a typical situation in which a page includes an `af:inputText` component, an `af:commandButton` component, and an `af:outputText` component. When the user enters a value for the `af:inputText`, then clicks the `af:commandButton`, the input value is reflected in the `af:outputText`. Without PPR, clicking the `af:commandButton` triggers a full-page refresh. Using PPR, you can limit the scale of the refresh to only those components you want to refresh, in this case the `af:outputText` component. To achieve this, you would do two things: ■ Set up the `af:commandButton` for partial submit by setting the `partialSubmit` attribute to `true`. Doing this causes the command component to start firing partial page requests each time it is clicked. ■ Define which components are to be refreshed when the partial submit takes place, in this example the `af:outputText` component, by setting the `partialTriggers` attribute for each of them to the id of the component triggering the refresh. In this example, this means setting the `partialTriggers` attribute of the `af:outputText` component to give the id of the `af:commandButton` component. The steps above achieve PPR using a command button to trigger the partial page refresh. The main reason why partial page rendering can significantly boost the performance is that full page refresh does not happen and the framework artifacts (such as ADF Rich Client JS library, and style sheets) are not reloaded and only a small part of page is refreshed. In several cases, this means no extra data is fetched or no geometry management. The ADF Rich Client has shown that partial page rendering results in the best client-side performance. Besides the impact on the client side, server-side processing can be faster and can have better server-side throughput and scalability.

Table 7–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Use ADF rich client navigation.	ADF Rich Client has an extensive support for navigation. One of the common use cases is tabbed navigation. This is currently supported by components like navigationPane which can bind to xmlMenuModel to easily define navigation. There is one drawback in this approach. It results in a full page refresh every time the user switches the tab. One option is to use panelTabbed instead. panelTabbed has built-in support for partial page rendering of the tabbed content without requiring any developer work. However, panelTabbed cannot bind to any navigational model and the content has to be available from within the page, so it has limited applicability.
Cache resources.	Developers are strongly encouraged to ensure that any resources that can be cached (images, CSS, JavaScript) have their cache headers specified appropriately. Also, client requests for missing resources on the server result in addition round trips to the server. To avoid this, make sure all the resources are present on the server.

Table 7–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description

Define custom styles at the top of the page.	A common developer task is to define custom styles inside a regular page or template page. Since most browsers use progressive scanning of the page, a late introduction of styles forces the browser to recompute the page. This impacts the page layout performance. For better performance, define styles at the top of the page and possibly wrap them inside the ADF group tag.
	
	An HTML page basically has two parts, the "head" and the "body". When you put an `af:document` component on your page, this component creates both parts of the page for you. Any child component of the `af:document` is in the "body" part of the page. To get a component (or static CDATA content) to show up in the "head", use the "metaContainer" facet.
	
	To get a component (or static CDATA content) to display in the "head", use the "metaContainer" facet as follows:
	
	`<af:document title="#{attrs.documentTitle}" theme="dark">`
	`<f:facet name="metaContainer">`
	`<af:group><![CDATA[`
	`<style type="text/css">`
	`.TabletNavigationGlobal {`
	`text-align: right;`
	`padding-left: 0px;`
	`padding-right: 10px;`
	`white-space: nowrap;`
	`}`
	`HTML[dir=rtl] .TabletNavigationGlobal {`
	`text-align: left;`
	`padding-left: 10px;`
	`padding-right: 0px;`
	`}`
	`</style>`
	`]]>`
	`<af:facetRef facetName="metaContainer"/>`
	`</af:group>`
	`</f:facet>`
	`<af:form ...>`
	`<af:facetRef facetName="body"/>`
	`</af:form>`
	`</af:document>`
	
	If you use page templates, consider including `af:document` and `af:form` in the template definition and expose anything that you may want to customize in those tags through the page template attributes and page template `af:facetRef`. Your templates are then able to utilize the metaContainer facet if they have template-specific styling as shown above. Also, your usage pages do not have to repeat the same document and form tags on every page.
	
	See the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* for details about `af:facetRef`.

Table 7–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Optimize custom JavaScript code.	ADF Rich Client uses JavaScript on the client side. The framework itself provides most of the functionality needed. However, you may have to write custom JavaScript code. To get the best performance, consider bundling the JavaScript code into one JS lib (one JavaScript file) and deliver it to the client. The easiest approach is to use the ADF tag: `<af:resource type="javascript" source=" "/>`. If most pages require custom JavaScript code, the tag should be included in the application template. Otherwise, including it in particular pages can result in better performance. If custom the JavaScript code lib file becomes too big, then consider splitting it into meaningful pieces and include only the pieces needed by the page.Overall, this approach is faster since the browser cache is used and the html content of the page is smaller.
Disable debug output mode.	The `debug-output` element in the `trinidad-config.xml` file specifies whether output should be more verbose to help with debugging. When set to `TRUE`, the output debugging mechanism in Trinidad produces pretty-printed, commented HTML content. To improve performance by reducing the output size, you should disable the debug output mode in production environments. Set the `debug-output` element to `FALSE`, or if necessary, remove it completely from the `trinidad-config.xml` file.
Disable test automation.	Enabling test automation parameter `oracle.adf.view.rich.automation.ENABLED` generates a client component for every component on the page which can negatively impact performance. Set the `oracle.adf.view.rich.automation.ENABLED` parameter value to `FALSE` (the default value) in the `web.xml` file to improve performance.
Disable animation.	ADF Rich Client framework has client side animation enabled by default. Animation is introduced to provide an enhanced user experience. Some of the components, like popup table, have animation set for some of the operations. While using animation can improve the user experience, it can increase the response time when an action is executed. If speed is the biggest concern, then animation can be disabled by setting the flag in `trinidad-config.xml`
Disable client-side assertions.	Assertions on client-side code base can have a significant impact on client-side performance. Set the parameter value to `FALSE` (the default value) to disable client-side assertions. Also ensure that the `oracle.adf.view.rich.ASSERT_ENABLED` is not explicitly set to `TRUE` in the `web.xml` file.
Disable JavaScript Profiler.	When the JavaScript `oracle.adf.view.rich.profiler.ENABLED` profiler is enabled, an extra round-trip occurs on every page in order to fetch the profiler data. Disable the profiler in the `web.xml` file to avoid this extra round-trip.
Disable resource debug mode.	When resource debug mode is enabled, the HTTP response headers do not tell the browser (or WebCache) that resources (JS libraries, CSS style sheets, or images) can be cached. Disable the `org.apache.myfaces.trinidad.resource.DEBUG` parameter in the `web.xml` file to ensure that caching is enabled.
Disable timestamp checking.	The `org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION` parameter controls whether jsp or jspx files are checked for modifications each time they are accessed. Ensure that the parameter value `org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION` is set to `FALSE` (the default value) in the web.xml file.

Table 7–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Disable checking for CSS file modifications.	The `org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION` parameter controls when CSS file modification checks are made. To aid in performance, this configuration option defaults to `false` - do not check for css file modifications. Set this to `TRUE` if you want the skinning css file changes to be reflected without stopping or starting the server.
Enable content compression.	By default, style classes that are rendered are compressed to reduce page size. In production environments, make sure you remove the `DISABLE_CONTENT_COMPRESSION` parameter from the `web.xml` file or set it to `FALSE`. For debugging, turn off the style class content compression. You can do this by setting the `DISABLE_CONTENT_COMPRESSION` property to `TRUE`.
Enable JavaScript obfuscation.	ADF Faces supports a run time option for providing a non-obfuscated version of the JavaScript library. The obfuscated version is supplied by default, but the non-obfuscated version is supplied for development builds. Obfuscation reduces the overall size of the JavaScript library by about 50%. To provide an obfuscated ADF Faces build, set the `org.apache.myfaces.trinidad.DEBUG_JAVASCRIPT` parameter to `FALSE` in the web.xml file. There are two ways to check that the code is obfuscated using Firefox with Firebug enabled: Check the download size: 1. Ensure that "All" or "JS" is selected on the Net tab. 2. Locate the "`all-11-version.js`" entry. 3. Check the size of the column. It should be about 1.3 MB (as opposed to 2.8 MB). Check the source: 1. From the Script tab select "all-11-*version*.js from the drop-down menu located above the tabs. 2. Examine the code. If there are comments and long variable names, the library is not obfuscated. Note: Copyright comments are kept even in the obfuscated version of the JS files.
Enable library partitioning.	In the Oracle 11*g* Release, library partitioning is on by default. In previous versions library partitioning was off by default. Ensure that the library partitioning is on by validating the `oracle.adf.view.rich.libraryPartitioning.DISABLED` property is set to `false` in the web.xml file.

7.2.3 Tuning ADF Faces Component Attributes

Table 7–3 provides configuration recommendations for ADF Faces Component Attributes:

Table 7–3 ADF Faces Component Attributes

Configuration Recommendation	Description
Use the "immediate" attribute.	ADF Rich Client components have an `immediate` attribute. If a component has its `immediate` attribute set to `TRUE` (`immediate="true"`), then the validation, conversion, and events associated with the component are processed during the applyRequestValues phase. These are some cases where setting `immediate` to `TRUE` can lead to better performance. ■ The commandNavigationItem in the navigationPane can use the `immediate` attribute set to `TRUE` to avoid processing the data from the current screen while navigating to the new page. ■ If the input component value has to be validated before the other values, `immediate` should be set to `TRUE`. In case of an error it be detected earlier in the cycle and additional processing be avoided. ADF Rich Client is built on top of JSF and uses standard JSF lifecycle. See "Understanding the JSF and ADF Faces Lifecycles" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*. There are some important issues associated with the `immediate` attribute. Refer to "Using the Immediate Attribute" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for more information. Note that this is an advanced feature. Most of the performance improvements can be achieved using the `af:subform` component. Refer to *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for `af:subform` details.
Use the "visible" and "rendered" attributes.	All ADF Faces Rich Client display components have two properties that dictate how the component is displayed on the page: ■ The `visible` property specifies simply whether the component is to be displayed on the page, or is to be hidden. ■ The `rendered` property specifies whether the component shall exist in the client page at all. The EL expression is commonly used to control these properties. For better performance, consider setting the component to not rendered instead of not visible, assuming there is no client interaction with the component. Making a component not rendered can improve server performance and client response time since the component does not have client side representation.

Table 7–3 (Cont.) ADF Faces Component Attributes

Configuration Recommendation	Description
Use client-side events.	ADF Rich Client framework provides the client-side event model based on component-level events rather than DOM level. The client-side event model is a very useful feature that can speed up the application. Review the following performance considerations: ■ Consider using client-side events for relatively simple event handling that can be done on the client side. This improves client side performance by reducing the number of server round trips. Also, it can increase server-side throughput and scalability since requests do not have to be handled by the server. ■ By default, the events generated on the client by the client components are propagated to the server. If a client-side event handler is provided, consider canceling the event at the end of processing so that the event does not propagate to the server.
Use the "id" attribute.	The "id" attribute should not be longer than 7 characters in length. This is particularly important for naming containers. A long id can impact performance as the amount of HTML that must be sent down to the client is impacted by the length of the ids.
Use client-side components.	ADF Rich Client framework has client-side components that play a role in client-side event handling and component behavior. The `clientComponent` attribute is used to configure when (or if) a client-side component should be generated. Setting `clientComponent` attribute to `TRUE` has a performance impact, so determine if its necessary to generate client-side components. For more information, see "Client-side Components" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.

7.2.4 Performance Considerations for Table and Tree Components

Table, Tree, and TreeTable are some of the most complex, and frequently used, components. Since these components can include large sets of data, they can be the common source of performance problems. Table 7–4 provides some performance recommendations.

Table 7–4 Table and Tree Component Configurations

Configuration Recommendation	Description
Modify table fetch size.	Tables have a fetch size which defines the number of rows to be sent to the client in one round-trip. To get the best performance, keep this number low while still allowing enough rows to fulfill the initial table view port. This ensures the best performance while eliminating extra server requests. In addition, consider keeping the table fetch size and iterator range size in sync. By default, the table fetch size is set to the EL expression `#{bindings.<name>.rangeSize}` and should be equal to the iterator size. For more information see "Using Tables and Trees" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.
Disable column stretching.	Columns in the table and treeTable components can be stretched so that there is no unused space between the end of the last column and the edge of the table or treeTable component. This feature is turned off by default due to potential performance impacts. Turning this feature on may have a performance impact on the client rendering time, so use caution when enabling this feature with complex tables. For more information see "Use Column Stretching" in *Oracle Fusion Applications Developer's Guide*.
Consider using header rows and frozen columns only when necessary.	The table component provides features that enable you to set the row Header and frozen columns. These options can provide a well-designed interface which can lead to a good user experience. However, they can impact client-side performance. To get the best performance for table components, use these options only when they are needed.

7.2.5 Performance Considerations for autoSuggest

`autoSuggest` is a feature that can be enabled for inputText, inputListOfValues, and inputComboboxListOfValues components. When the user types characters in the input field, the component displays a list of suggested items. The feature performs a query in the database table to filter the results. In order to speed up database processing, a database index should be created on the column for which autosuggest is enabled. This improves the component's response times especially when the database table has a large number of rows.

7.2.6 Data Delivery - Lazy versus Immediate

Data for Table, Tree, and other stamped components can be delivered immediately or lazily. By default, lazy delivery is used. This means that data is not delivered in the initial response from the server. Rather, after the initial page is rendered, the client asks the server for the data and gets it as a response to the second request.

In the case of immediate delivery, data can be in line with the response to the page request. It is important to note that data delivery is per component and not per page. This means that these two can be mixed on the same page.

When choosing between these two options, consider the following:

Lazy Delivery (default)	Lazy delivery should be used on pages where content is not immediately visible unless the user scrolls down to it. In this case the time to deliver the visible context to the client be shorter, and the user perceives better performance. Lazy delivery is implemented using data streaming technique. The advantage of this approach is that the server has the ability to execute data fetches in parallel and stream data back to the client as soon as the data is available. The technique performs very well for a page with two tables, one that returns data very quickly and one that returns data very slowly. Users see the data for the fast table as soon as the data is available. Executing data fetches in parallel also speeds up the total time to fetch data. This gives an advantage to lazy loading for the cases of multiple and possible slow data fetches.
Immediate Delivery	Immediate delivery (`contentDelivery="immediate"`) should be used if table data control is fast, or if it returns a small set of data. In these cases the response time be faster than using lazy delivery. Another advantage of immediate delivery is less server resource usage, compared to lazy delivery. Immediate delivery sends only one request to the server, which results in lower CPU and memory usage on the server for the given user interaction.

7.2.7 Performance Considerations for DVT Components

DVT components are data visualization components built on top of ADF Rich Client components. DVT components include graphs, gauges, Gantt charts, pivot tables and maps. Table 7–5 provides some configuration recommendations for DVT components:

Table 7–5 DVT Component Configurations

Configuration Recommendation	Description
Modify the RangeSize attribute.	The RangeSize attribute defines the number of rows to return simultaneously. A RangeSize value of -1 causes the iterator to return all the rows. Using a lower value may improve performance, but it may be harder to stop the data and any data beyond rangeSize is not available in the view.
Use horizontal text instead of vertical text.	By default, pivot tables use horizontal text for column headers. However, there is an option to use vertical text as well. Vertical text can be used by specifying a CSS style for the header format such as: `writing-mode:tb-rl;filter:flipV flipH;` While vertical text can look better in some cases, it has a performance impact when the Firefox browser is used. The problem is that vertical text is not native in Firefox as it is in Internet Explorer. To show vertical text, the pivot table uses images produced by GaugeServlet. These images cannot be cached as the text is dynamic and depends on the binding value. Due to this, every rendering of the pivot table incurs extra round-trips to the server to fetch the images, which impact network traffic, server memory, and CPU. To have the best performance, consider using horizontal text instead of vertical text.

7.3 ADF Server Performance

Oracle ADF Server components consist of the non-UI components within ADF. These include the ADF implementations of the model layer (ADFm), business services layer (ADFbc), and controller layer (ADFc). As the server components are highly configurable, it is important to choose the combination of configurations that best suits the available resources with the specified application performance and functionality.

7.3.1 View Objects Tuning

View objects (VOs) provide many tuning options to enable a developer to tailor the View Object to the application's specific needs. View Objects should be configured to use the minimal feature set required to fulfill the functional requirement. The *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* provides detailed information on tuning View Objects. Provided here are some tips pertaining to View Object performance.

7.3.1.1 Creating View Objects

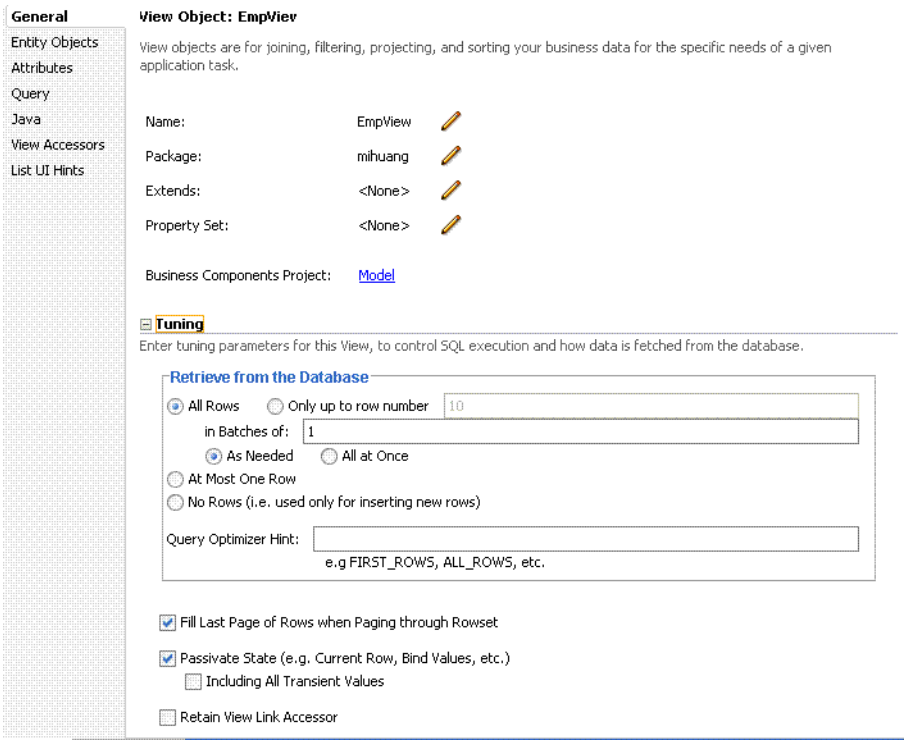
To maximize View Object performance, the View Object should match the intended usage. For instance, data retrieved for a list of values pick-list is typically read-only, so a read-only View Object should be used to query this data. Tailoring the View Object to the specific needs of the application can improve performance, memory usage, CPU usage, and network usage.

View Object Type	Description
Read-only View Objects	Consider using a read-only View Object if the View Object does not have to insert or update data. There are two options for read-only View Objects: ■ Non-updatable EO-based View Objects ■ Expert-mode View Objects Non-updatable EO-based View Objects offer the advantage of a customizable select list at run time which retrieve attributes needed in the UI, data reads from local cache (instead of re-executing a database query), and data consistency with other updatable View Objects based on the same EO. Expert-mode View Objects have the ability to perform SQL operations not supported by EOs and avoid the small performance impact from coordinating View Object and EO rows. EO-based View Objects can be marked non-updatable by deselecting the "updatable" option in the selected EO for the View Object, which can also be done by adding the parameter `ReadOnly="true"` on the `EntityUsage` attribute in the View Object XML definition.
Insert-only View Objects	For View Objects that are used only for inserting records, you can prevent unnecessary select queries from being executed when using the View Object. To do this, set the option No Rows in the Retrieve from the Database group box in the View Objects Overview tab. This sets MaxFetchSize to 0 (zero) for the View Object definition.
run time-created View Objects	View Objects can be created at run time using the `createViewObjectFromQueryStmt()` API on the AM. However, avoid using run time-created View Objects unless absolutely necessary due to potential performance impacts and complexity of tuning.

7.3.1.2 Configuring View Object Data Fetching

View Object performance is largely dependent on how the view object is configured to fetch data. If the fetch options are not tuned correctly for the application, then the view object may fetch an excessive amount of data or may take too many round-trips to the database. Fetch options can be configured through the **Retrieve from the Database** group box in the View Object dialog Figure 7–1.

Figure 7–1 View Object Dialog



Fetch Option	Description
Fetch Mode	The default fetch option is the All Rows option, which is retrieved as needed (`FetchMode="FETCH_AS_NEEDED"`) or all at once (`FetchMode="FETCH_ALL"`), depending on which option is appropriate. The As Needed option ensures that an `executeQuery()` operation on the view object initially retrieves only as many rows as necessary to fill the first page of a display. The number of rows is set based on the view object's range size.
Fetch Size	In conjunction with the fetch mode option, the Batches field controls the number of records fetched simultaneously from the database (FetchSize in the View Object, XML). The default value is 1, which may impact performance unless only 1 row is fetched. The suggested configuration is to set this value to *n*+1 where *n* is the number of rows to be displayed in the user interface. Note that for DVT objects, Fetch Size should be *n*+1 where *n* is either rangeSize or the likely maximum rowset size if rangeSize is -1.
Max Fetch Size	The default max fetch size for a View Object is -1, which means that there is no limit to the number of rows the View Object can fetch. Setting a max fetch size of 0 (zero) makes the View Object insert-only. In cases where the result set should only contain *n* rows of data, the option Only Up to Row Number should be selected and set or call `setMaxFetchSize(N)` to set this programmatically. To set this manually, add the parameter `MaxFetchSize` to the View Object XML. For View Objects whose WHERE clause expects to retrieve a single row, set the option At Most One Row. This option ensures that the view object knows not to expect any more rows and skips its normal test for that situation. In this case no select query is issued and no rows are fetched. Max fetch size can also be used to limit the impact from an non-selective query that may return hundreds (or thousands) of rows. In such cases, specifying the max fetch size limits the number of rows that can be fetched and stored into memory.

Fetch Option	Description
Forward-Only Mode	If a data set is only traversed going forward, then forward-only mode can help performance when iterating through the data set. This can be configured by programmatically calling `setForwardOnly(true)` on the View Object. Setting forward-only can also prevent caching previous sets of rows as the data set is traversed.

7.3.1.3 Additional View Object Configurations

Table 7–6 provides additional tuning considerations when using the View Object:

Table 7–6 Additional View Object Configurations

Configuration Recommendation	Description
Optimize large data sets.	View Objects provide a mechanism to page through large data sets so that a user can jump to a specific page in the results. This is configured by calling `setRangeSize(N)` followed by `setAccessMode(RowSet.RANGE_PAGING)` on the View Object where N is the number of rows contained within 1 page. When navigating to a specific page in the data set, the application can call `scrollToRangePage(P)` on the View Object to navigate to page P. Range paging fetches and caches only the current page of rows in the View Object row cache at the cost of another query execution to retrieve each page of data. Range paging is not appropriate where it is beneficial to have all fetched rows in the View Object row cache (for example, when the application must read all rows in a data set for an LOV or page back and forth in records of a small data set).
Disable "spillover" configurations when possible.	You can use the data source as "virtual memory" when the JVM container runs out of memory. By default this is disabled and can be enabled (if needed) by setting `jbo.use.pers.coll=true`. Keep this option disabled (if possible) to avoid a potential performance impact.
Review SQL style configuration.	If the generic SQL92 SQL style is used to connect to generic SQL92-compliant database, then some View Object tuning options do not apply. The View Object fetch size is one such tuning option. When SQL92 SQL style is used, the fetch size defaults to 10 rows, regardless of what is configured for the View Object. The SQL style is set when defining the database connection. By default when defining an Oracle database connection, the SQL style can be `Oracle`. To manually override the SQL style, pass the parameter `-Djbo.SQLBuilder="SQL92"` to the JVM at startup.
Use bind variables for view object queries.	If the query associated with the View Object contains values that may change from execution to execution, consider using bind variables. This may help to avoid re-parsing the query on the database. Bind variables can be added to the View Object in the Query section of the View Object definition.
Use query optimizer hints for view object queries.	The View Object can pass hints to the database to influence which execution plan to use for the associated query. The optimizer hints can be specified in the Retrieve from the Database group box.
Use dynamic SQL generation.	View Objects can be configured to dynamically generate SQL statements at run time instead of defining the SQL at design time. A View Object instance, configured with generating SQL statements dynamically, can avoid re-querying a database. This is especially true during page navigation if a subset of all attributes with the same key EO list is used in the subsequent page navigation. Performance can be improved by activating a superset of all the required attributes to eliminate a subsequent query execution.

7.3.2 Batch Processing

Batch processing enables multiple inserts, updates, and deletes to be processed together when sending the operations to the database. Enabling this feature is done on the EO by either selecting the "Use Update Batching" check box in the Tuning section

of the EO's General tab, or by directly modifying the EO's XML file and adding the parameter `BatchThreshold` with the specified batch size to the `Entity` attribute.

The `BatchThreshold` value is the threshold at which a group of operations can be batched instead of performing each operation one at a time. If the threshold is not exceeded, then rows may be affected one at a time. On the other hand, more rows than specified by the threshold can be batched into a single batch.

Note that the `BatchThreshold` configuration for the EO is not compatible if an attribute in the EO exists with the configuration to refresh after insert (`RetrievedOnInsert="true"`) or update (`RetrievedOnUpdate="true"`).

7.3.3 RangeSize Tuning

This parameter controls the number of records ADFm requests from the BC layer simultaneously. The default `RangeSize` is 25 records. Consider setting this value to the number of records to be displayed in the UI simultaneously for the View Object so that the number of round-trips between the model and BC layers is reduced to one. This is configured in the `Iterator` attribute of the corresponding page's page definition XML.

7.3.4 Application Module Design Considerations

Designing an application's module granularity is an important consideration that can significantly impact performance and scalability. It is important to not that each root application module generally holds its own database connection. If a user session consumes multiple root application modules, then that user session can potentially hold multiple database connections simultaneously. This can occur even if the connections are not actively being used due to the general affinity maintained between an application module and a user session. To reduce the possibility that a user can hold multiple connections at once, consider the following options:

- Design larger application modules to encompass all of the functionality that a user needs.
- Nest smaller application modules under a single root application module so that the same database connection can be shared among the nested application modules.

More information can be found in the "What You May Need to Know About Application Module Granularity" and "Defining Nested Application Modules" sections of *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

7.3.5 Application Module Pooling

Application module (AM) pooling enables multiple users to share several application module instances. The configurations for the AM pool vary depending on the expected usage of the application. For detailed explanations of the different AM pool configurations, see " Tuning Application Module Pools" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

Most of the AM pool parameters can be set through Oracle JDeveloper. The configurations are saved in `bc4j.xcfg`, which can be manually edited if needed. Parameters can also be set at the system level by specifying these as JVM parameters (`-D`property=value). The `bc4j.xcfg` configuration takes precedence over the JVM configuration; this enables a generic system-level configuration to be overridden by an application-specific exception.

Table 7–7 Application Module (AM) Pool Tuning

Configuration Recommendation	Description
Optimize the number of AM pools in the application.	Parameters applied at the system level are applied per AM pool. If the application uses more than 1 AM pool, then system-level values for the number of AM instances must be multiplied by the number of AM pools to realize the actual limits specified on the system as a whole. For instance, if an application uses 4 separate AM pools to service the application and a system-level configuration is used to limit the max AM pool size to 100, then this can result in a maximum of 400 AM instances (4 pools * 100 max pool size). If the intent is to limit the entire application to a max pool size of 100, then the system-level configuration should specify a max pool size of 25 (100 max pool size / 4 pools). Finer granularity for configuring each AM pool can be achieved by configuring each pool separately through JDev or directly in bc4j.xcfg.
Optimize the number of database connections.	By default AM instances retain their database connections even when checked back into the AM pool. There are many performance benefits to maintain this association. To maintain performance, consider configuring more AM instances than the maximum number of specified database connections. NOTE: If you have an AM pool that needs to be used as root pool, consider tuning at the specific AM pool level. For pools that are infrequently used, consider tuning pool sizes on the pool level so that top-level application parameters are not used. For more information see "Setting Pool Configuration Parameters" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

7.3.5.1 General AM Pool Configurations

The following guidelines can be used as a general starting point when tuning AM and AM pool behavior. Details for each parameter can be found in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*. More specific tuning for memory or CPU usage can be found in Section 7.3.5.2, "AM Pool Sizing Configurations".

Table 7–8 AM Pool Tuning Parameters

Parameter	Description
`jbo.ampool.initpoolsize`	Specifies the number of application module instances to create when the pool is initialized (default is zero). Setting a nonzero initial pool size increases the time to initialize the application, but improves subsequent performance for operations requiring an AM instance. A general guideline is to configure this to 10% more than the anticipated number of concurrent AM instances required to service all users.
`jbo.ampool.maxpoolsize`	Specifies the maximum number of application module instances that the pool can allocate (default is 4096). The pool can never create more application module instances than this limit imposes. A general guideline is to configure this to 20% more than the initial pool size to allow for some additional growth.
`jbo.ampool.minavailablesize`	Specifies the minimum number of available application module instances that the pool monitor should leave in the pool during a resource cleanup operation (default is 5). The ideal minimum value for this configuration should be at least 1 to avoid the costs of re-creating the AM pool. Setting this to zero (0) can cause the pool itself to be cleaned up when all instances have been idle for longer than the idle time out.

Table 7–8 (Cont.) AM Pool Tuning Parameters

Parameter	Description
`jbo.ampool.maxavailablesize`	Specifies the ideal maximum number of application module instances in the pool when not under abnormal load (default is 25). When the pool monitor wakes up to do resource cleanup, it tries to remove available application module instances to bring the total number of available instances down to this ideal maximum. Instances that have not been used for a period longer than the idle instance time out is cleaned up at this time, and then additional available instances can be removed if necessary to bring the number of available instances down to this size.
`jbo.recyclethreshold`	Specifies the maximum number of application module instances in the pool that attempt to preserve session affinity for the next request made by the session that used them last before releasing them to the pool in managed-state mode (default is 10). The referenced pool size should always be less than or equal to the maximum pool size. This enables the configured number of available instances to try and remain "loyal" to the affinity they have with the most recent session that released them in managed state mode. A general guideline is to configure this to the expected number of concurrent users that perform multiple operations with short think times. If there are no users expected to use the application with short think times, then this can be configured to 0 (zero) to eliminate affinity.
`jbo.ampool.timetolive`	Specifies the number of milliseconds that an application module instance lives in the pool. After this time, the instance is a candidate for removal during the next resource cleanup regardless of whether it would bring the number of instances in the pool below `minavailablesize` The default is 3600000ms or 1 hour. The default value is sufficient for most applications.
`jbo.ampool.maxinactiveage`	Specifies the number of milliseconds after which to consider an inactive application module instance in the pool as a candidate for removal during the next resource cleanup (default is 600000ms = 10 minutes).
`jbo.ampool.monitorsleepinterval`	Specifies the length of time in milliseconds between pool resource cleanup (default is 600000ms = 10 minutes). While the number of application module instances in the pool should never exceed the maximum pool size, available instances that are candidates for removal from the pool do not get "cleaned up" until the next time the application module pool monitor wakes up to do its job.
`jbo.dofailover`	Specifies whether to disable or enable failover. By default, failover is disabled. To enable failover, set the parameter to `true`. With failover enabled, the state information is automatically passed when the AM is checked back into the AM pool. This enables any other AM instance to activate the state at any time.
`jbo.locking.mode`	Specifies the locking mode (`optimistic` or `pessimistic`). The default is `pessimistic`, which means that a pending transaction state can be created on the database with row-level locks. With pessimistic locking mode, each time an AM is recycled, a rollback is issued in the JDBC connection. Web applications should set the locking mode to `optimistic` to avoid creating the row-level locks.
`jbo.doconnectionpooling`	Specifies whether the AM instance can be disconnected from the database connection when the AM instance is returned to the AM pool. This enables an application to size the AM pool larger than the database connection pool. The default is `false`, which means that an AM instance can retain its database connection when the AM instance is returned to the AM pool. When set to `true`, the AM can release the database connection back to the database connection pool when the AM instance is returned to the AM pool. Note that before an AM is disconnected from the database connection, a rollback can be issued on that database connection to revert any pending database state.
`jbo.txn.disconnect_level`	When used in conjunction with `jbo.doconnectionpooling=true`, specifies BC4J behavior for maintaining JDBC ResultSets. By default `jbo.txn.disconnect_level` is 0, and passivation can be used to close any open ResultSets when the database connection is disconnected from the AM instance. Configuring `jbo.txn.disconnect_level` to 1 can prevent this behavior to avoid the passivation costs for this situation.

7.3.5.2 AM Pool Sizing Configurations

The following AM pool sizing parameters control the AM pool size. Consider adjusting these values to tune memory or CPU usage.

For parameters that can be configured for memory-constrained systems, see Table 7–9.

Table 7–9 AM Pool Sizing Configurations - Memory Considerations

Parameter	Description
`jbo.ampool.initpoolsize`	Set this to a low value to conserve memory at the cost of slower performance when additional AM instances are required. The default value of 0 (zero) does not create any AM instances when the AM pool is initialized.
`jbo.ampool.maxpoolsize`	Configure this to prevent the number of AM instance from exceeding the determined value. However, if this is set too low, then some users may see an error accessing the application if no AM instances are available.
`jbo.ampool.minavailablesize`	Set to 0 (zero) to shrink the pool to contain no instances when all instances have been idle for longer than the idle time out after a resource cleanup. However, a setting of 1 is commonly used to avoid the costs of re-creating the AM pool.
`jbo.ampool.maxavailablesize`	Configure this to leave the maximum number of available instances specified after a resource cleanup.

For parameters that can be configured to reduce the load on the CPU to some extent through a few parameters, see Table 7–10.

Table 7–10 AM Pool Sizing Configurations - CPU Considerations

Parameter	Description
`jbo.ampool.initpoolsize`	Set this value to the number of AM instances you want the application pool to start with. Creating AM instances during initialization takes the CPU processing costs of creating AM instances during the initialization instead of on-demand when additional AM instances are required.
`jbo.recyclethreshold`	Configure this value to maintain the AM instance's affinity to a user's session. Maintaining this affinity as much as possible save the CPU processing cost of needing to switch an AM instance from one user session to another.

7.3.5.3 AM Pool Resource Cleanup Configurations

These parameters affect the frequency and characteristics for AM pool resource cleanups. Details about resource cleanup can be found in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

For memory-constrained systems, configure the AM pool to clean up more AM instances more frequently so that the memory consumed by the AM instance can be freed for other purposes. However, reducing the number of available AM instances and increasing the frequency of cleanups can result in higher CPU usage and longer response times. See Table 7–11 for more information.

Table 7–11 AM Pool Resource Cleanup Configurations - Memory Considerations

Parameter	Description
`jbo.ampool.minavailablesize`	A setting of 0 (zero) shrinks the pool to contain no instances when all instances have been idle for longer than the idle time out. However, a setting of 1 is commonly used to avoid the costs of re-creating the AM pool
`jbo.ampool.maxavailablesize`	A lower value generally results in more AM instances being removed from the pool on a cleanup.
`jbo.ampool.timetolive`	A lower value reduces the time an AM instance can exist before it must be removed at the next resource cleanup.
`jbo.ampool.maxinactiveage`	A low value results in more AM instances being marked as a candidate for removal at the next resource cleanup.
`jbo.ampool.monitorsleepinterval`	This controls how frequent resource cleanups can be triggered. Configuring a lower interval results in inactive AM instances being removed more frequently to save memory.

The AM pool can be configured to reduce the need for CPU processing by allowing more AM instances to exist in the pool for longer periods of time. This generally comes at the cost of consuming more memory.

Table 7–12 AM Pool Resource Cleanup Configurations - CPU Considerations

Parameter	Description
`jbo.ampool.minavailablesize` and `jbo.ampool.maxavailablesize`	Setting these to a higher value leaves more idle instances in the pool, so that AM instances do not have to be recreated at a later time. However, the values should not be set excessively high to keep more AM instances than can be required at maximum load.
`jbo.ampool.timetolive`	A higher value increases the time an AM instance can exist before it must be removed at the next resource cleanup.
`jbo.ampool.maxinactiveage`	A higher value results in fewer AM instances being marked as a candidate for removal at the next resource cleanup.
`jbo.ampool.monitorsleepinterval`	Configuring a higher interval results in less frequent resource cleanups.

7.3.6 ADFc: Region Usage

Adding regions to a page can be a powerful addition to the application. However, regions can be a resource-intensive component on the page. For better performance, consider using regions only when the specific functionality is required.

7.3.7 Reusing Static Data

If the application contains static data that can be reused across the application, the cache data can be collected using a shared application module. More information on creating and using shared application modules can be found in "Sharing Application Module View Instances" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

7.3.8 Conditional Validations

For resource-intensive validations on entity attributes, consider using preconditions to selectively apply the validations only when needed. The cost of validation must be weighted against the cost of the precondition to determine if the precondition is beneficial to the performance. More information on specifying preconditions for

validation can be found in "How to Set Preconditions for Validation" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

8

Oracle TopLink (EclipseLink) JPA Performance Tuning

This chapter describes some of the available performance tuning features for EclipseLink, an open-source persistence framework used with Oracle TopLink. The chapter includes the following topics:

- Section 8.1, "About Oracle TopLink and EclipseLink"
- Section 8.2, "Efficient SQL Statements and Queries"
- Section 8.3, "Cache Configuration Tuning"
- Section 8.4, "Coherence Integration"
- Section 8.5, "Mapping and Descriptor Configurations"
- Section 8.6, "Analyzing EclipseLink JPA Entity Performance"

Note: For more information on performance tuning in these areas, see the following:

- EclipseLink Performance Tuning at `http://wiki.eclipse.org/EclipseLink/Performance`
- EclipseLink JPA Tuning Best Practices at `http://wiki.eclipse.org/EclipseLink/FAQ/JPA/BestPractices`
- Introduction to Optimization at `http://wiki.eclipse.org/Optimizing_the_EclipseLink_Application_(ELUG)#Introduction_to_Optimization`
- Optimizing for a Production Environment at `http://wiki.eclipse.org/Optimizing_the_EclipseLink_Application_(ELUG)#Optimizing_for_a_Production_Environment`.

8.1 About Oracle TopLink and EclipseLink

Oracle TopLink includes the open source EclipseLink as the Java Persistence API (JPA) implementation. Oracle TopLink extends EclipseLink with advanced integration into the Oracle Application Server.

The Java Persistence API (JPA) is a specification for persistence in Java EE and Java SE applications. In JPA, a persistent class is referred to as an entity. An entity is a plain old

Java object (POJO) class that is mapped to the database and configured for usage through JPA using annotations, persistence XML, or both. This chapter focuses on tuning JPA in the context of EJB3.0 and a Java EE environment.

The information in this chapter assumes that you are familiar with the basic functionality of EclipseLink. Before you begin tuning, consider reviewing the introductory information found at the following:

- "Introduction to Java Persistence API" section of the EclipseLink Developer's Guide at http://wiki.eclipse.org/Introduction_to_Java_Persistence_API_(ELUG)
- "Introduction to EclipseLink JPA" section of EclipseLink Developer's Guide at http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_%28ELUG%29
- "Considering JPA Entity Architecture" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Considering_JPA_Entity_Architecture
- Introduction to EclipseLink Queries at http://wiki.eclipse.org/Introduction_to_EclipseLink_Queries_(ELUG)
- Introduction to Cache at http://wiki.eclipse.org/Introduction_to_Cache_(ELUG)
- Introduction to Mapping and Configuration at http://wiki.eclipse.org/Introduction_to_EclipseLink_Mapping_and_Configuration_(ELUG)

For more information on Oracle TopLink, see the TopLink page on OTN http://www.oracle.com/technology/products/ias/toplink/index.html .

[Note that as of Oracle TopLink Release 11*g*, the older Toplink APIs have been deprecated. For more information, see the TopLink Release Notes at http://www.oracle.com/technology/products/ias/toplink/doc/11110/relnotes/toplink-relnotes.html#CHDGAEDJ]

> **Note:** This chapter serves as a 'quick start' guide to performance tuning JPA in the context of a Java EE environment. While the chapter provides common performance tuning considerations and related documentation resources, it is not meant to be comprehensive list of areas to tune.

8.2 Efficient SQL Statements and Queries

This section covers using efficient SQL statements and SQL querying. Table 8–1 and Table 8–2 show tuning parameters and performance recommendations related to SQL statements and querying.

Table 8–1 EJB/JPA Using Efficient SQL Statements and Querying

Tuning Parameter	Description	Performance Notes
Parameterized SQL Binding	Using parameterized SQL and prepared statement caching, you can improve performance by reducing the number of times the database SQL engine parses and prepares SQL for a frequently called query. EclipseLink enables parameterized SQL by default. However, not all databases and JDBC drivers support these options. Note that the Oracle JDBC driver bundled with Oracle Application Server does support this option. The persistence property in persistence.xml "eclipselink.jdbc.bind-parameters" is used to configure this. See Also: "Using EclipseLink JPA Extensions - Bind Parameters" at `http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Bind_Parameters` Default Value: PERSISTENCE_UNIT_DEFAULT (which is true by default)	Leave parameterized SQL binding enabled for selected databases and JDBC drivers that support these options.
JDBC Statement Caching	Statement caching is used to lower the performance impact of repeated cursor creation and repeated statement parsing and creation; this can improve performance for applications using a database. Note: For J2EE applications, use the data source's statement caching (and do not use EclipseLink Statement Caching for EJB3.0/JPA, for example: `eclipselink.jdbc.cache-statements"="true"`). Set this option in an Oracle Weblogic data-source by setting `Statement Cached Type` and `Statement Cached Size` configuration options. See also "Increasing Performance with the Statement Cache" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*. Default Value: The Oracle Weblogic Server data source default statement cache size is 10 statements per connection.	You should always enable statement caching if your JDBC driver supports this option. The Oracle JDBC driver supports this option.

Table 8–1 (Cont.) EJB/JPA Using Efficient SQL Statements and Querying

Tuning Parameter	Description	Performance Notes
Fetch Size	The JDBC fetch size gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed. For large queries that return a large number of objects, you can configure the row fetch size used in the query to improve performance by reducing the number database hits required to satisfy the selection criteria. See Also: "Using EclipseLink JPA Extensions - Fetch Size" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Bind_Parameters Most JDBC drivers use a default fetch size of 10. If you are reading 1000 objects, increasing the fetch size to 256 can significantly reduce the time required to fetch the query's results. Note: The default value means use the JDBC driver default value, which is typically 10 rows for the Oracle JDBC driver. To configure this, use query hint `"eclipselink.jdbc.fetch-size"`. Default Value: 0	The optimal fetch size is not always obvious. Usually, a fetch size of one half or one quarter of the total expected result size is optimal. Note that if you are unsure of the result set size, incorrectly setting a fetch size too large or too small can decrease performance.
Batch Writing	Batch writing can improve database performance by sending groups of `INSERT`, `UPDATE`, and `DELETE` statements to the database in a single transaction, rather than individually. The persistence property in persistence.xml `"eclipselink.jdbc.batch-writing"="JDBC"` is used to configure this. See Also: "How to Use Batch Writing for Optimization" at http://wiki.eclipse.org/Optimizing_the_EclipseLink_Application_(ELUG)#How_to_Use_Batch_Writing_for_Optimizatio Default Value: Off	Enable for the persistence unit.
Change Tracking	This is an optimization feature that lets you tune the way EclipseLink detects changes in an Entity. See Also: "Using EclipseLink JPA Extensions for Tracking Changes" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Using_EclipseLink_JPA_Extensions_for_Tracking_Changes Default Value: AttributeLevel if using weaving (J2EE default), otherwise Deferred.	Leave at default AttributeLevel for best performance.
Weaving	Can disable through persistence.xml properties `"eclipselink.weaving"` Default Value: On	Leave on for best performance.

Table 8–1 (Cont.) EJB/JPA Using Efficient SQL Statements and Querying

Tuning Parameter	Description	Performance Notes
Read Only	Setting an EJB3.0 JPA Entity to read-only ensures that the entity cannot be modified and enables EclipseLink to optimize unit of work performance. Set through query hint "eclipselink.read-only". Can also be set at entity level using `@ReadOnly` class annotation. See Also: "Using EclipseLink JPA Extensions - Read Only" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Read_OnlyUsing_EclipseLink_JPA_Extension Default Value: False	For optimal performance use read-only on any query where the resulting objects are not changed.
firstResult and maxRows	These are JPA query properties that are used for paging large queries. Typically, these properties can be used when the entire result set of a query returning a large number of rows is not needed. For example, when a user scans the result set (a page at a time) looking for a particular result and then discards the rest of the data after the record is found. See Also: "How to Use Result Set Pagination" at http://wiki.eclipse.org/Optimizing_the_EclipseLink_Application_(ELUG)#How_to_Use_Result_Set_Pagination_for_Optimization	Use on queries that can have a large result set and only a subset of the objects is needed.
Sequence number pre-allocation	Sequence number pre-allocation enables a batch of ids to be queried from the database simultaneously in order to avoid accessing the database for an id on every insert. See Also: "Sequencing and Pre-allocation Size" at http://wiki.eclipse.org/Optimizing_the_EclipseLink_Application_(ELUG)#Sequence_Number_Preallocation Default Value: 50	Always use sequence number pre-allocation for best performance for inserts. `SEQUENCE` or `TABLE` sequencing should be used for optimal performance, not `IDENTITY` which does not allow pre-allocation.

8.2.1 Entity Relationships Query Parameter Tuning

Table 8–2 shows the Entity relationship query parameters for performance tuning.

Table 8–2 EJB3.0 Entity Relationship Query Performance Options

Tuning Parameter	Description	Performance Notes
Batch Reading	The eclipselink.batch hint supplies EclipseLink with batching information so subsequent queries of related objects can be optimized in batches instead of being retrieved one-by-one or in one large joined read. Batching is only allowed on queries that have a single object in their select clause.The query hint to configure this is "eclipselink.batch". See Also: EclipseLink User's Guide section "Using EclipseLink JPA Extensions - Batch " at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Batch Default Value: Off	Use for queries of tables with columns mappings to table data you need.You should only use either batch-reading or joining if you know that you are going to access all of the data; if you do not intend to access the relationships, then just let indirection defer their loading. Batch reading is more efficient than joining because it avoids reading duplicate data; therefore for best performance for queries where batch reading is supported, consider using batch reading instead of join reading.
Join	Join reading is a query optimization feature that enables a single query for a class to return the data to build the instances of that class and its related objects. Use this feature to improve query performance by reducing database access. By default, relationships are not join-read: each relationship is fetched separately when accessed if you are using lazy-loading, or as a separate database query if you are not using lazy-loading. You can specify the use of join in `JPQL (JOIN FETCH)`, or you can set it multi-level in a query hint, "eclipselink.join-fetch". It also can be set in the mapping annotation @JoinFetch. Joining is part of the JPA specification, whereas batch reading is not. And, joining works on queries that not work with batch reading. For example, joining works on queries with multiple objects in the select clause, queries with a single result, and for cursors and first/max results, whereas batch reading does not. See Also: "Using EclipseLink JPA Extensions - Join Fetch " at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Join_Fetch Default Value: Not Used	Use for queries of tables with columns mappings to table data you need.You should only use either batch-reading or joining if you know that you are going to access all of the data; if you do not intend to access the relationships, then just let indirection defer their loading.For the best performance of selects, where batch reading is not supported, a join is recommended

Table 8–2 (Cont.) EJB3.0 Entity Relationship Query Performance Options

Tuning Parameter	Description	Performance Notes
Lazy loading	Without lazy loading on, when EclipseLink retrieves a persistent object, it retrieves all of the dependent objects to which it refers. When you configure lazy reading (also known as indirection, lazy loading, or just-in-time reading) for an attribute mapped with a relationship mapping, EclipseLink uses an indirection object as a place holder for the referenced object. EclipseLink defers reading the dependent object until you access that specific attribute. This can result in a significant performance improvement, especially if the application is interested only in the contents of the retrieved object, rather than the objects to which it is related. See Also: "What You May Need to Know About EclipseLink JPA Lazy Loading" at `http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#What_You_May_Need_to_Know_About_EclipseL` Default Value: On for collection mapping (ToMany mappings, @OneToMany, @ManyToMany) Default Value: Off for reference (ToOne mappings, @OneToOne, @ManyToOne) (Note that setting lazy loading On for @OneToOne, @ManyToOne requires weaving, which is On by default for J2EE.)	Use lazy loading for all mappings. Using lazy loading and querying the referenced objects using batch reading or Join is more efficient than Eager loading.

8.3 Cache Configuration Tuning

This section describes tuning the default internal cache that is provided by EclipseLink. Oracle Toplink/EclipseLink can also be integrated with Oracle Coherence. For information on configuring and tuning an EclipseLink Entity Cache using Oracle Coherence, see Section 8.4, "Coherence Integration".

The default settings for EJB3.0/JPA used with the EclipseLink persistence manager and cache are no locking, no cache refresh, and cache-usage `DoNotCheckCache`. To ensure that your application uses the cache and does not read stale data from the cache (when you do not have exclusive access), you must configure these and other isolation related settings appropriately. Table 8–3 shows the cache configuration options.

Note: By default, EclipseLink assumes that your application has exclusive access to the data it is using (that is, there are no external, non-EclipseLink, applications modifying the data). If your application does not have exclusive access to the data, then you must change some of the defaults from Table 8–3.

Table 8–3 EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Object Cache	EclipseLink sessions provide an object cache. EJB3.0 JPA applications that use the EclipseLink persistence manager create EclipseLink sessions that by default use this cache. This cache, known as the **session cache**, retains information about objects that are read from or written to the database, and is a key element for improving the performance of an EclipseLink application. Typically, a server session's object cache is shared by all client sessions acquired from it. Isolated sessions provide their own session cache isolated from the shared object cache. If you do not wish to cache an object you should set shared to false (in `@Cache` or persistence.xml (`eclipselink.cache.shared.default`,`eclipselink.cache.shared.<ENTITY>`)). See Also: "Session Cache" at http://wiki.eclipse.org/Introduction_to_Cache_(ELUG)#Session_Cache "How to Use the Persistence Unit Properties for Caching" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Table_19-13. Default Value: Enabled (shared is True)	Generally it is recommended that you leave caching enabled. If you have an object that is always read from the database, as in a pessimistic locked object, then the cache for that entity should be disabled. For example: `set shared=false`. Also, consider disabling the cache for infrequently accessed entities

Query Result Set Cache	In addition to the object cache in EclipseLink, EclipseLink also supports a query cache:
	■ The object cache indexes objects by their primary key, allowing primary key queries to obtain cache hits. By using the object cache, queries that access the data source can avoid the cost of building the objects and their relationships if the object is already present.
	■ The query cache is distinct from the object cache. The query cache is indexed by the query and the query parameters - not the object's primary key. This enables any query executed with the same parameters to obtain a query cache hit and return the same result set.
	The query hints for a query cache are:
	`"eclipselink.query-cache"`
	`"eclipselink.query-cache.size"`
	`"eclipselink.query-cache.invalidation"`
	See Also: "How to Cache Query Results in the Query Cache" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Queries_%28ELUG%29#How_to_Cache_Query_Results_in_the_Query_ and "How to Use EclipseLink JPA Query Hints" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#How_to_Use_EclipseLink_JPA_Query_Hints
	Default Value: Not Used
		Use for frequently executed non-primary key queries with infrequently changing result sets.Use with a cache invalidation time out to refresh as needed.

Table 8–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes

Cache Size	Cache size can be configured through persistence properties:
	`"eclipselink.cache.size.<entity>"`
	`"eclipselink.cache.size.default"`
	`"eclipselink.cache.type.default"`
	See Also: "Using EclipseLink JPA Extensions for Caching" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Table_19-13
	"JPA Best Practices - Configure the Cache" at http://wiki.eclipse.org/EclipseLink/FAQ/JPA/BestPractices#M3._Configure_the_Cache
	Default Value: Type `SoftWeak`, Size 100 (per Entity).
		Set the cache size relative to how much memory you have available, how many instances of the class you have, the frequency the entities are accessed, and how much caching you want based on your tolerance for stale data.
		Consider creating larger cache sizes for entities that have many instances that are frequently accessed and stale data is not a big issue.
		Consider using smaller cache sizes or no cache for frequently updated entities that must always have fresh data, or infrequently accessed entities.

Locking	Oracle supports the locking policies shown in Table 8–4: no locking, optimistic, pessimistic, and read-only. Locking is set through JPA @Version annotation, eclipselink.read-only See Also: "Introduction to EclipseLink Application Development - "Locking" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Locking "Using EclipseLink JPA Extensions Pessimistic Lock" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Pessimistic_Locking How to Use EclipseLink Locking at http://wiki.eclipse.org/EclipseLink/Examples/JPA/Locking "Configuring Locking" at http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_(ELUG)#Configuring_Locking Default Value: No Locking	For entities that can be updated concurrently, consider using the locking policy to prevent a user from writing over another users changes. To optimize performance for read-only entities, consider defining the entity as read-only or use a read-only query hint.

Table 8–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Cache Usage	By default, all query types search the database first and then synchronize with the cache. Unless refresh has been set on the query, the cached objects can be returned without being refreshed from the database. You can specify whether a given query runs against the in-memory cache, the database, or both. To get performance gains by avoiding the database lookup for objects already in the cache, you can configure that the search attempts to retrieve the required object from the cache first, and then search the data source only if the object is not in the cache. For a query that looks for a single object based on a primary key, this is done by setting the query hint "`eclipselink.cache-usage`" to `CheckCacheByExactPrimaryKey`. See Also: "Using EclipseLink JPA Extensions - Cache Usage" at `http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Cache_Usage` Default Value: `DoNotCheckCache`	For faster performance on primary key queries, where the data is typically in the cache and does not require a lot of refreshing, it is recommended to check the cache first on these queries (using `CheckCacheByExactPrimaryKey`). This avoids the default behavior of retrieving the object from the database first and then for objects already in the cache, returning the cached values (not updated from the database access, unless refresh has been set on the query).
Isolation	There is not a single tuning parameter that sets a particular database transaction isolation level in a JPA application that uses EclipseLink. In a typical EJB3.0 JPA application, a variety of factors affect when database transaction isolation levels apply and to what extent a particular database transaction isolation can be achieved, including the following: ■ Locking mode ■ Use of the Session Cache ■ External Applications ■ Database Login method `setTransactionIsolation` See Also: "Database Transaction Isolation Levels" at `http://wiki.eclipse.org/Using_Advanced_Unit_of_Work_API_(ELUG)#Database_Transaction_Isolation_Levels`	

Table 8–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Cache Refreshing	By default, EclipseLink caches objects read from a data source. Subsequent queries for these objects access the cache and thus improve performance by reducing data source access and avoiding the cost of rebuilding object's and their relationships. Even if a query accesses the data source, if the objects corresponding to the records returned are in the cache, EclipseLink uses the cached objects. This default caching policy can lead to stale data in the application. Refreshing can be enabled at the entity level (`alwaysRefresh` or `refreshOnlyIfNewer` and `expiry`) and at the query level (with the `eclipselink.refresh` query hint). You can also force queries to go to the database with (`disableHits`). Using an appropriate locking policy is the only way to ensure that stale or conflicting data does not get committed to the database. For more information see: Section 8.3.1, "Cache Refreshing Scenarios" See Also: EclipseLink User's Guide section, "Configuring Cache Refreshing" and "How to Use the @Cache_Annotation" Default Value: No Cache Refreshing	Try to avoid entity level cache refresh and instead, consider configuring the following: ▪ cache refresh on a query-by-query basis ▪ cache expiration ▪ isolated caching

8.3.1 Cache Refreshing Scenarios

There are a few scenarios to consider for data refreshing in the cache, all with performance implications:

- In the case where you never want cached data and always want fresh data, consider using an isolated cache (Shared=False). This is the case when certain data in the application changes so frequently that it is desirable to always refresh the data, instead of only refreshing the data when a conflict is detected.
- In the case when you want to avoid stale data, but getting stale data is not a major issue, then using a cache expiry policy would be the recommended solution. In this case you should also use optimistic locking, which automatically refresh stale objects when a locking error occurs. If using optimistic locking, you could also enable the entity @Cache attributes `alwaysRefresh` and `refreshOnlyIfNewer` to allow queries that access the database to refresh any stale objects returned, and avoid refreshing invalid objects when unchanged. You may also want to enable refreshing on certain query operations when you know you want refreshed data, or even provide the option of refreshing something from the client that would call a refreshing query.
- In the case when you are not concerned about stale data, you should use optimistic locking; this automatically refresh stale objects in the cache on locking errors.

8.3.2 Locking Modes

The locking modes, as shown in Table 8–4, along with EclipseLink cache-usage and query refreshing options, ensures data consistency for EJB entities using JPA. The different combinations have both functional and performance implications, but often

the functional requirements for up-to-date data and data consistency lead to the settings for these options, even when it may be at the expense of performance.

For more information see "Configuring Locking" at http://wiki.eclipse.org/Introduction_to_EclipseLink_JPA_(ELUG)#Configuring_Locking.

Table 8–4 Locking Mode Policies

Locking Option	Description	Performance Notes
No Locking	The application does not prevent users overwriting each other's changes. This is the default locking mode. Use this mode if the Entity is never updated concurrently or concurrent reads and updates to the same rows with read-committed semantics is sufficient. See Also: Introduction to EclipseLink Application Development at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Locking Default Value: No Locking	In general, no locking is faster, but may not meet your needs for data consistency.
Optimistic	All users have read access to the data. When a user attempts to make a change, the application checks to ensure the data has not changed since the user read the data. See Also: "Introduction to EclipseLink Application Development: Locking" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Locking	If infrequent concurrent updates to the same rows are expected, then optimistic locking may provide the best performance while providing data consistency guarantees.
Pessimistic	The first user who accesses the data with the purpose of updating it locks the data until completing the update. See Also: "Introduction to EclipseLink Application Development: Locking" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Locking	If frequent concurrent updates to the same rows are expected, pessimistic locking may be faster than optimistic locking that is getting a lot of concurrent access exceptions and retries. When using pessimistic locking at the entity level, it is recommended that you use it with an isolated cache (Shared=False) for best performance.
Read Only	Setting an EJB3.0 JPA Entity to read-only ensures that the entity cannot be modified and enables EclipseLink to optimize unit of work performance. Set at the entity level using @ReadOnly class annotation. Can also be set at the query level through query hint "`eclipselink.read-only`". See Also: "Introduction to EclipseLink Application Development: Locking" at http://wiki.eclipse.org/Introduction_to_EclipseLink_Application_Development_(ELUG)#Locking Using EclipseLink JPA Extension - Read Only at http://wiki.eclipse.org/Http:/wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Read_OnlyUsing_EclipseLink_JPA_Extension.	Defining an entity as read-only can perform better than an entity that is not defined as read-only, yet does no inserts, updates, or deletes, since it enables EclipseLink to optimize the unit of work performance. Always use read-only for all read-only operations

8.4 Coherence Integration

Oracle Toplink can be integrated with Oracle Coherence. This integration is provided through the Oracle TopLink Grid feature. With TopLink Grid, there are several types of integration with EclipseLink JPA features.

For example:

- Replace the default EclipseLink L2 cache with Coherence. This provides support for very large L2 caches that span cluster nodes. EclipseLink's default L2 cache improves performance for multi-threaded and Java EE server hosted applications running in a single JVM, and requires configuring special cache coordination features if used across a cluster.
- Configure entities to execute queries in the Coherence data grid instead of the database. This allows clustered application deployments to scale beyond database-bound operations.

For more information on using EclipseLink JPA with a Coherence Cache, see "JPA on the Grid" Approach at http://www.oracle.com/technology/products/ias/toplink/doc/11110/grid/tlgug003.htm

For more information on Oracle Toplink integration with Oracle Coherence, see "Oracle TopLink Integration with Coherence Grid Guide" at http://www.oracle.com/technology/products/ias/toplink/doc/11110/grid/toc.htm

8.5 Mapping and Descriptor Configurations

EclipseLink can transform data between an object representation and a representation specific to a data source. This transformation is called mapping and it is the core of a EclipseLink project.

A mapping corresponds to a single data member of a domain object. It associates the object data member with its data source representation and defines the means of performing the two-way conversion between object and data source.

For information on Mapping see, "Optimizing Mappings and Descriptors" in the EclipseLink User Guide at http://wiki.eclipse.org/Optimizing_the_EclipseLink_Application_(ELUG)#Optimizing_Mappings_and_Descriptors.

For more information on Descriptors see, "Configuring Common Descriptor Options" at http://wiki.eclipse.org/Configuring_a_Descriptor_(ELUG)#Configuring_Common_Descriptor_Options

8.6 Analyzing EclipseLink JPA Entity Performance

This section lists a few features in EclipseLink that can help you analyze your JPA application performance:

- For profiling performance, see "Measuring EclipseLink Performance with the EclipseLink Profiler" in the EclipseLink User's Guide. Note that this tool is intended for use with single-threaded finite use cases.
- For debugging performance issues and testing, you can view the SQL generated from EclipseLink. To view the SQL, increase the logging level to "FINE" by using the EclipseLink JPA extensions for logging.

For more information about setting the log level, see "Using EclipseLink JPA Extensions for Logging" at http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG)#Using_EclipseLink_JPA_Extensions_for_Logging

For best performance, remember to restore the logging levels to the default levels when you are done profiling or debugging.

9

Oracle Web Cache Performance Tuning

This chapter provides guidelines for improving the performance of Oracle Web Cache.

- Section 9.1, "About Oracle Web Cache"
- Section 9.2, "Optimizing Hardware Resources"
- Section 9.3, "Optimizing Network Connections"
- Section 9.4, "Optimizing Platform Connections"
- Section 9.5, "Increasing Cache Hit Rates"
- Section 9.6, "Optimizing Response Time"
- Section 9.7, "Optimizing Performance with Oracle ADF"

9.1 About Oracle Web Cache

Oracle Web Cache is a content-aware server accelerator, or a reverse proxy, for the Web tier.

Oracle Web Cache is the primary caching mechanism provided with Oracle Fusion Middleware. Caching improves the performance, scalability, and availability of Web sites that run on Oracle Fusion Middleware by storing frequently accessed URLs in memory. It can also improve the performance, scalability, and availability of Web sites that run on any Web server or application server, such as Oracle HTTP Server and Oracle WebLogic Server.

For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

9.2 Optimizing Hardware Resources

- Hardware Resources
- Memory Configuration

9.2.1 Hardware Resources

Oracle Web Cache performs best with one very powerful CPU or two CPUs. Because Oracle Web Cache is an in-memory cache, it is rarely limited by CPU cycles. Additional CPUs do not increase performance significantly. However, the speed of the processors is critical-use the fastest CPUs you can afford. Use more CPUs if Web Cache is sharing the system with other Oracle application server components or other applications.

Note that Oracle Web Cache is limited by the available addressable memory. Additional memory can increase performance and scalability. For information about the amount of memory needed, see Section 9.2.2, "Memory Configuration".

Oracle Web Cache has two processes: one for the administration server and one for the cache server.

- The administration server process is used for configuring and monitoring Oracle Web Cache. This process consumes very little CPU time. However, when viewing the statistics pages in Oracle Web Cache Manager, the administration server process must query the cache server process to obtain the relevant metrics. Accessing the statistics pages frequently, or setting a high refresh rate on a statistics page can affect cache server performance.
- The cache server process uses three threads: one to manage the front-end activities, a second to manage the back-end activities, and a third to process requests.

For a cost-effective way to use Oracle Web Cache, run it on a fast two-CPU dedicated computer with lots of memory. See the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache* for information about various deployment scenarios.

For a Web site with more than one Oracle Web Cache instance, consider installing each instance on a separate two-CPU node, either as part of a cache cluster or as a standalone instance. When Oracle Web Cache instances are on separate nodes, you are less likely to encounter operating system limitations, particularly in network throughput. For example, two caches on two separate two-CPU nodes are less likely to encounter operating system limitations than two caches on one four-CPU node.

Of course, if other resources are competing with Oracle Web Cache for CPU usage, you should take the requirements of those resources into account when determining the number of CPUs needed. Although a separate node for Oracle Web Cache is optimal, you can also derive a significant performance benefit from Oracle Web Cache running on the same node as the rest of the application Web server.

9.2.2 Memory Configuration

To avoid swapping documents in and out of the cache, configure enough memory for the cache. Generally, the amount of memory (maximum cache size) for Oracle Web Cache should be set to at least **512** MB. Your application's memory requirements can vary based upon factors such document size, number of documents, the number of HTTP headers returned, and whether ESI is present. To get a close approximation on the maximum amount of memory required, you may apply the formula provided below. If your application uses ESI then all templates and document fragments must be accounted for when figuring the TotalDocs and the AvgDocSize.

Estimated Cache size in bytes = 1.25 *(TotalDocs * ((AvgDocSize/8192+1) *8192+ 16384))

- 0.25 accounts for the run time memory usage. The Web Cache action limit is set to 5% below than the maximum Web Cache size by default. Web Cache also allocates 5% of the total cache size to optimize access misses that cannot be cached.
- TotalDocs refers to the total number of documents you intend to place in Web Cache.
- The AvgDocSize is self-explained.
- Remember to convert the estimated cache size is returned in bytes by the formula.

The memory formula presented above was verified against actual memory usage measurements and it showed very close results as can be seen in the table below:

Number of Cached	Doc Size In	Measured Cache	Formula Generated Results
Docs	Bytes	Size in MB	Size in MB
3300.00	102400	499.61	499.51
5525.00	51200	499.27	499.08
11050.00	51200	998.54	998.17
6600.00	102400	999.22	999.02
13200.00	102400	1998.44	1998.05
22100.00	51200	1997.07	1996.34
3300.00	102400	499.61	499.51

9.2.2.1 Configuring WebCache Memory

The cache is empty when Oracle Web Cache starts. For monitoring to be valid, ensure that the cache is fully populated. That is, ensure that the cache has received enough requests so that a representative number of documents are cached.

The Oracle Web Cache Statistics page (Monitoring > Web Cache Statistics) provides information about the current memory use, the maximum memory use and the total documents currently resident in Oracle Web Cache. Note the following metrics in the Cache Overview table:

- Size of Documents in Cache shows the current logical size of the cache, which is the size of the valid documents in the cache. For example, if the cache contains two documents, one 3 KB and one 50 KB, the Size of Documents in Cache is 53 KB, the total of the two sizes.
- Configured Maximum Cache Size indicates the maximum cache size as specified in the Resource Limits page.
- Current Allocated Memory displays the physical size of the cache, which is the amount of data memory allocated by Oracle Web Cache for cache storage and operation. This number is always smaller than the process size shown by operating system statistics because the Oracle Web Cache process, like any user process, consumes memory in other ways, such as instruction storage, stack data, thread, and library data.
- Current Action Limit is 95% of the Configured Maximum Cache Size. This number is usually larger than the Current Allocated Memory.

If the Current Allocated Memory is greater than the Current Action Limit, Oracle Web Cache begins to use allocated but unused memory, and may begin garbage collection to free more memory. During garbage collection, Oracle Web Cache removes the less popular and less valid documents from the cache in favor of the more popular and more valid documents to obtain space for new HTTP responses without exceeding the maximum cache size.

If the Current Allocated Memory is close to or greater than the Current Action Limit, increase the maximum cache size to avoid swapping documents in and out of the cache. For more information, see "Specifying Properties for an Oracle Web Cache System Component" in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

9.3 Optimizing Network Connections

- Network Bandwidth
- Network Connections
- Network-Related Parameters

9.3.1 Network Bandwidth

When you use Oracle Web Cache, ensure that each system has sufficient network bandwidth to accommodate the throughput load. Otherwise, the network may be saturated but Oracle Web Cache has additional capacity. For example, if an application generates 100 megabits of data or more per second, 10/100 Megabit Ethernet can be saturated.

If the network is saturated, consider using Gigabit Ethernet rather than 10/100 Megabit Ethernet. Gigabit Ethernet provides the most efficient deployment scenario to avoid network collisions, retransmissions, and bandwidth starvation. Additionally, consider using two separate network cards: one for incoming client requests and one for requests from the cache to the application Web server.

Use network-monitoring utilities that show network bandwidth usage. If the network is under utilized and throughput is less than expected, check whether the CPUs are saturated.

9.3.2 Network Connections

It is important to specify a reasonable number for the maximum connection limit for the Oracle Web Cache server. If you set a number that is too high, performance can be affected, resulting in slower response time. If you set a number that is too low, fewer requests can be satisfied. Strike a balance between response time and the number of requests processed concurrently.

To help determine a reasonable number, consider the following factors:

- The maximum number of clients that you intend to serve concurrently at any given time.
- The average size of a document and the average number of requests per document.
- Network bandwidth. The amount of data that can be transferred at any one time is limited by the network bandwidth.
- The percentage of cache misses. Cache misses are forwarded to the application Web server. Those requests consume additional network bandwidth, resulting in longer response times; especially if a large percentage of requests are cache misses.
- How quickly a document is processed. Use a network monitoring utility, such as `ttcp` or LoadRunner to determine how quickly your system processes a document.
- The cache cluster member capacity, if you have a cache cluster environment. The capacity reflects the number of incoming connections from other cache cluster members. Set the cluster member capacity using the Clustering page (Properties > Clustering) of Oracle Web Cache Manager.

WARNING: Do not set the values listed above to an arbitrarily high value. **Oracle Web Cache** sets aside some resources such as memory for each connection. Altering these values can adversely affect performance.

Use various tools, such as those available with the operating system and with Oracle Web Cache, to help determine the maximum number of connections. For example, the `netstat-a` command enables you to determine the number of established connections; the `ttcp` utility enables you to determine how fast a document is processed. The Oracle Web Cache Manager provides statistics on hits and misses.

For detailed instructions on how to set the maximum number of incoming connections, see "Specifying Properties for an Oracle Web Cache System Component" in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

9.3.3 Network-Related Parameters

Besides the number of network connections, other network-related parameters for Oracle Web Cache, the application Web server, and the operating system can affect response time. In most situations, the default settings are sufficient.

If response time is slow, you should tune Oracle Web Cache, the application Web server, and operating system parameters that affect connections, as explained in this section.

For Oracle Web Cache, check the values of the following settings:

- Keep-Alive Timeout

  The amount of time a network connection is left open after Oracle Web Cache sends a response to a browser. Keep-Alive enables an HTTP client to send multiple requests to Oracle Web Cache using the same network connection. By default, the connection is left open for five seconds, which is typically enough time for the browser to send subsequent requests to Oracle Web Cache using the same connection.

  If the network between the browser and Oracle Web Cache is slow, consider increasing the timeout, experiment with 10 seconds then 20 seconds and perhaps up to 30 seconds.

  If you receive the following error, either increase the maximum incoming connections for Oracle Web Cache or lower the Keep-Alive Timeout:

  ```
  11313: The cache server reached the maximum number of allowed incoming
  connections. Listening is temporarily suspended.
  ```

  With a heavy load, such as during stress-testing, if clients continuously send one request and then disconnect, set the Keep-Alive Timeout to 0. With this value, Oracle Web Cache closes the connection as soon as the request is completed, to free up resources.

  Set the Keep-Alive Timeout value in the Network Timeouts page (Properties > Network Timeouts).

- Origin Server Timeout

  The amount of time for the application Web server to generate a response to Oracle Web Cache. If the application Web server or proxy server is unable to

generate a response within that time, Oracle Web Cache sends a network apology page to the browser.

Usually, this value should be equal to the response time of the slowest document served by the application Web Server. If the value is too low, long-running requests can timeout before the response is complete. If the value is too high and the application Web server hangs for some reason, it can take longer for Oracle Web Cache to failover to another application Web server.

Set this value in the Network Timeouts page (Properties > Network Timeouts).

For the application Web server, check the values of the following settings in the application Web server's configuration file (`httpd.conf`). (These particular parameter names are specific to the Oracle HTTP Server.)

- `KeepAlive`

  Whether to allow persistent connections. Persistent connections allow a client to send multiple sequential requests through the same connection.

  Make sure `KeepAlive` is enabled. This can improve performance because the connection is set up only once and is kept open for subsequent requests from the same client.

  - `KeepAliveTimeout`: The time a connection is left open to wait for the next request from the same client. If requests are primarily from Oracle Web Cache, you can set this value fairly high. A reasonable value is 30 seconds.
  - `MaxKeepAliveRequests`: The maximum number of requests to allow during a persistent connection. Set to 0 to allow an unlimited number of requests.
  - `MaxClients`: The maximum number of clients that can connect to the application Web server simultaneously.

  If `KeepAlive` is enabled for the application Web server, you may require more concurrent httpd server processes, and you may have to set the `MaxClients` directive to a higher value.

  If client requests have a short response time, you may be able to improve performance by setting `MaxClients` to a lower value. However, when the `MaxClients` value is reached, no additional processes can be created, causing other requests to fail. The `MaxClients` limit on the application Web server should be greater than or equal to the application Web server capacity as set through the Oracle Web Cache Manager.

  For the operating system, check the TCP time-wait setting. This setting controls the amount of time that the operating system holds a port, not allowing new connections to use the same port.

  On the Linux operating system, validate the value of /proc/sys/net/ipv4/tcp_fin_timeout. On the Solaris Operating System, check the `tcp_time_wait_interval` setting, using the following command:

  ```
  ndd -get /dev/tcp tcp_time_wait_interval.
  ```

  On Windows, check the value of `TcpTimeWaitDelay` in the following key in the registry:

  ```
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
  ```

  This setting is usually only an issue during stress testing, if you continuously open more TCP/IP connections from one client computer. In this situation, lower the

TCP time-wait setting. In real world deployments, this is rarely an issue because it is unlikely that a single client can generate a huge number of connections.

9.4 Optimizing Platform Connections

- UNIX Connections
- Windows Connections

9.4.1 UNIX Connections

On most UNIX platforms, each client connection requires a separate file descriptor. The Oracle Web Cache server attempts to reserve the maximum number of file descriptors when it starts. If you have root privileges, you can increase this number. For example, for the LINUX Operating System you can increase the maximum number of file descriptors by modifying Oracle Web Cache users file descriptors limits in /etc/security/limits.conf.

For example to allow the user "WC_USER" to have 4092 connections, in the /etc/security/limits.conf file add the following entries:

```
WC_User soft nofile 4092
WC_User hard nofile 4092
```

Ensure that there are adequate file descriptors available to any process on the host by increasing the `fs.file-max parameter` in the /etc/sysctl.conf file.

On Solaris Operating System you can increase the maximum number of file descriptors by setting the `rlim_fd_max` parameter. If `webcached` is not run as `root`, the Oracle Web Cache server logs an error message and fails to start.

9.4.2 Windows Connections

On Windows, only available kernel resources limit the number of file handles as well as socket handles - the size of paged and non-paged pools. However, the number of TCP ports the system can open restricts the number of active TCP/IP connections.

For more information on establishing connections, see "Set Resource Limits and Network Thresholds" in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

9.5 Increasing Cache Hit Rates

A *cache hit* is a web browser request that can be satisfied from documents stored in the cache. A *cache miss* is a web browser request that cannot be satisfied from documents stored in the cache and must be forwarded to the application web server.

If the ratio of cache hits to cache misses is low, consider the following ways to raise the cache hit rate:

- Use cookies and URL parameters to increase cache hit rates.

  Oracle Web Cache can cache different versions of a document with the same URL, based on request cookies or headers. To use this feature, applications may need to implement simple changes, such as creating a cookie or header that differentiates the documents.

  Some applications contain insignificant URL parameters, which can lead to different URLs representing the same content. If the documents are cached under their full URLs, the cache hit/miss ratio becomes very low. You can configure

Oracle Web Cache to ignore the non-differentiating URL parameter values, so that a single document is cached for different URLs, greatly increasing cache hit rates.

Sometimes the content for a set of documents is nearly identical. For example, the documents may contain hyperlinks composed of the same URL parameters with different session-specific values, or they may include some personalized strings in the document text, such as welcome greetings or shopping cart totals. You can configure Oracle Web Cache to store a single copy of the document with placeholders for the embedded URL parameters or the personalized strings, and to dynamically substitute the correct values for the placeholders when serving the document to clients.

For more information on multiple version documents, sessions, ignoring URL parameter values, and simple personalization, see "Getting Started with Administering Oracle Web Cache" in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

- Use redirection to cache entry documents.

  For some popular site entry documents, such as "/", that typically require session establishment, session establishment effectively makes the document non-cacheable to all new users without a session. To cache these documents while preserving session establishment, you can either:

  – Create a blank document that provides session establishment for all initial requests and redirects to the actual popular document. Subsequent redirected requests to the popular document can specify the session, enabling the popular document to be served from the cache.

  – Use a JavaScript that sets a session cookie for the popular documents.

  Note: For more information on configuring caching rules for documents requiring session establishment, see "Caching and Compressing Content" in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

- Use partial page caching where possible.

  Many Web documents, such as pages generated by OracleAS Portal, are composed of fragments with unique caching properties. For these pages, full-page caching is not feasible. However, Oracle Web Cache provides partial page caching using Edge Side Includes (ESI). With ESI, you can divide each Web page into a template and multiple fragments that can, in turn, be further divided into templates and lower level fragments. Each fragment or template is stored and managed independently; a full page is assembled from the underlying fragments upon request. Fragments can be shared among different templates, so that common fragments are not duplicated to waste cache space. Sharing can also greatly reduce the number of updates required when fragments expire.

  Depending on the application, updating a fragment can be cheaper than updating a full page. In addition, each template or fragment can have its own unique caching policies such as expiration, validation, and invalidation, so that each fragment in a full Web page can be cached if possible, even when some fragments are not cached or are cached for a much shorter period of time.

- Use ESI variables for improved cache hit/miss ratio for personalized pages.

  Personalized information often appears in Web pages, making them unique for each user. For example, many Web pages contain tens or hundreds of hyperlinks

embedding application session IDs. To resolve this, create your ESI pages with variables. Because variables can resolve to different pieces of request information or response information, the uniqueness of templates and fragments can be significantly reduced. This, in turn, results in better cache hit/miss ratios.

9.6 Optimizing Response Time

If you have not configured the application Web server or the cache correctly, response time may be slower than anticipated. This section summarizes much of the information presented in this chapter.

If the application Web server is responding more slowly than expected or if the application Web server is not responding to requests from the cache because it has reached its capacity, check the application Web server and Oracle Web Cache settings.

First, check the following:

- Caching rules: Ensure that you are caching the appropriate objects. Are there popular objects that you should cache but are not caching? Use the Popular Requests page (Monitoring > Popular Requests) to see a list of the most popular requests and to check that those objects are being cached.
- Priority rankings of the caching rules: Give frequently accessed non-cacheable documents a higher priority than cacheable documents. Give frequently accessed cacheable documents the lowest priority. Note that parsing of caching rules may be resource-intensive if a large number of rules are defined.
- Compression: If the network is a bottleneck for the client, compressing documents as they are cached can relieve some of the congestion on the network because compressed documents are smaller.

Then, check the following:

The application Web server configuration, particularly the `MaxClients`, `KeepAlive`, `KeepAliveTimeout`, and `MaxKeepAliveRequests` settings.

The `MaxClients` limit on the application Web server should be greater than or equal to the application Web server capacity as set through the Oracle Web Cache Manager.

The application Web server capacity as set using the Origin Servers page (Origin Servers, Sites, and Load Balancing > Origin Servers) of the Oracle Web Cache Manager. See the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache* for information about setting application Web server capacity.

Then, if the application Web server is still busier than anticipated, it may mean that the cache cannot process the requests and is routing more requests to the application Web server. Check the following Oracle Web Cache settings in the Oracle Web Cache Manager:

- The number of cache connections. Check Maximum Incoming Connections in the Resource Limits page (Properties > Resource Limits).
- The memory size for the cache. Check Maximum Cache Size in the Resource Limits page (Properties > Resource Limits).
- The cache cluster capacity. In a cache cluster, if cluster capacity is too low, a cache may not receive a response for owned content from a peer cache in the specified interval. As a result, the request is sent to the application Web server. Check Capacity in the Clustering page (Properties > Clustering). See the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache* for more information.

If the settings for the application Web server and Oracle Web Cache are set correctly, but the response times are still higher than expected, check system resources, especially:

- Network bandwidth
- CPU usage

9.7 Optimizing Performance with Oracle ADF

Consider the following configuration options for optimizing Oracle Web Cache performance with Oracle ADF Rich Client Applications:

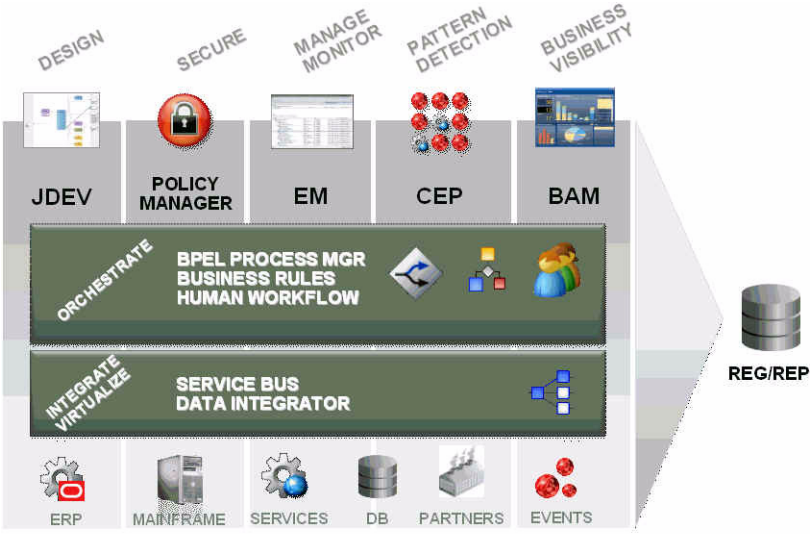
- After you configure the Maximum Cache Size setting in the Resource Limits page of Oracle Web Cache Manager, use a simulated load or an actual load to monitor the cache to see how much memory is actually used. Verify that any additional memory usage does not result in the host swapping memory to disk, as this may impact performance.
- Personalization and compression rules for all sites include the following:
  - Images should be cached but not compressed
  - CSS files should be both cached and compressed for all request types
  - JS files should be both cached and compressed for all request types
  - HTML files should be both cached and compressed
  - SWF files should be cached but not compressed
  - Add a rule to compress but not cache .jspx files for all GET and POSTS
  - Add a rule to compress but not cache \.jspx.*$ files for all GET and POSTS
  - Add a rule to compress but not cache adw\.jspx for all request types
  - Add a rule not to compress and not cache profiling.js for all request types

For more detail on setting cache and compression rules, see "Caching and Compressing Content," in *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

Part IV

SOA Suite Components

This part describes configuring Oracle Service-Oriented Architecture (SOA) Suite components to improve performance. Oracle SOA Suite is a component of Oracle Fusion Middleware. Oracle SOA Suite provides a complete set of service infrastructure components for designing, deploying, and managing SOA composite applications. The image below shows the Oracle SOA Platform.



Oracle SOA Suite enables services to be created, managed, and orchestrated into SOA composite applications. Composites enable you to easily assemble multiple technology components into one SOA composite application. SOA composite applications consist of:

- Service components: Service components are the basic building blocks of SOA composite applications. Service components implement a part of the overall business logic of the SOA composite application. BPEL Process, Oracle Mediator, Human task flow and decision services are examples of the service components.
- Binding components: Binding components connect SOA composite applications to external services, applications, and technologies. Binding components are organized into two groups:
  - Services: Provide the outside world with an entry point to the SOA composite application. The WSDL file of the service advertises its capabilities to external applications. The service bindings define how a SOA composite service can be invoked (for example, through SOAP).

  - References: Enable messages to be sent from the SOA composite application to external services (for example, the same functionality that partner links provide for BPEL processes, but at the higher SOA composite application level).

The SOA Suite Components are documented in the following chapters:

- Chapter 10, "Cross Component Tuning for SOA Suite"
- Chapter 11, "Oracle BPEL Process Manager Performance Tuning"
- Chapter 12, "Oracle Mediator Performance Tuning"
- Chapter 13, "Oracle Human Workflow Performance Tuning"
- Chapter 14, "Oracle Adapters Performance Tuning"
- Chapter 15, "Oracle Business Activity Monitoring Performance Tuning"
- Chapter 16, "User Messaging Service Performance Tuning"

10

Cross Component Tuning for SOA Suite

This chapter describes tuning configurations that can apply to multiple SOA Suite applications.

- Section 10.1, "About SOA Suite Configuration Properties"
- Section 10.2, "SOA Infrastructure Configurations"
- Section 10.3, "Modifying SOA Configuration Parameters"
- Section 10.4, "JVM Tuning Parameters"
- Section 10.5, "Database Settings"

For more information on any of the SOA Suite Applications, see Section IV, "SOA Suite Components" for a list of the application-specific documentation provided in this guide.

10.1 About SOA Suite Configuration Properties

This section provides additional resources for the SOA Suite components. Refer to the following sections of the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite* for more information on configuring the SOA Applications:

BPEL Properties: For setting the audit trail size, maximum document size for a variable, payload validation for incoming and outgoing messages, audit trail level, dispatcher thread level for invoke messages, system thread level, and engine thread level.

For more information, see, "Configuring BPEL Process Service Engine Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

Mediator Properties: For setting the audit level, metrics level, number of parallel worker threads, number of maximum rows retrieved for parallel processing, parallel thread sleep values, error thread sleep values, container ID refresh time, and container ID lease timeout values.

For more information, see "Introduction to Configuring Oracle Mediator" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

Workflow Notification Properties: For setting the workflow service notification mode and actionable e-mail address value.

For more information, see, "Configuring Human Workflow Notification Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

Workflow Task Service Properties: For setting the actionable e-mail account, adding the worklist application URL, selecting the pushback assignee, adding portal realm mapping, and adding the task auto release configuration priority.

For more information, see "Configuring Human Workflow Task Service Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

10.2 SOA Infrastructure Configurations

SOA Infrastructure configuration parameters impact the entire SOA Infrastructure. The following configurations are modified through the SOA-INFRA component:

- Viewing and setting the SOA Infrastructure audit level
- Capturing the state of the SOA composite application instance
- Enabling the payload validation of incoming messages
- Specifying the callback server and server URLs
- Setting UDDI registry properties
- Viewing the data source JNDI locations
- Setting the non-fatal connection retry count
- Setting Web service binding properties

For more information on SOA configuration, see "Configuring SOA Infrastructure Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

10.2.1 Audit Level

The Audit Level property enables you to select the level of information to be collected by the message tracking infrastructure. This information is collected in the instance data store (database) associated with the SOA Infrastructure. This setting has no impact on what is written to log files.

Value	Description
Off	No composite instance tracking and payload tracking information is collected. No more composite instances can be created. No logging is performed. Note that no logging and display of instances in Oracle Enterprise Manager Fusion Middleware Control Console can result in a slight performance increase for processing instances. Instances are created, but are not displayed.
Production	Composite instance tracking is collected, but the Oracle Mediator service engine does not collect payload details and the process service engine does not collect payload details for assign activities (payload details for other activities are collected). This level is optimal for most normal production operations.
Development	Enables both composite instance tracking and payload detail tracking. However, this setting may impact performance. This level is useful largely for testing and debugging purposes.

10.2.2 Composite Instance State

You can use the `CompositeInstanceStateEnabled` property to configure the SOA composite application instance state. Note, however, that enabling this option may impact performance during instance processing. This option enables separate tracking of the running instances. All instances are captured as either running or not running.

This information displays later in the State column of the composite instances tables for the SOA Infrastructure and SOA composite application. The valid states are running, completed, faulted, recovery needed, stale, terminated, suspended, and state not available.

10.2.3 Logging Level

The default logging level is "NOTIFICATION". For stress testing and production environments, consider using the lowest acceptable logging level, such as "ERROR" or "WARNING" whenever possible.

For more information on setting the logging levels for your applications, see "Configuring Log File" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

10.3 Modifying SOA Configuration Parameters

SOA and SOA-INFRA configurations are modifiable either through WLST or Oracle Enterprise Manager. To use WLST, use the following location:

<WLST_ ROOT>
/oracle.as.soainfra.config/oracle.as.soainfra.config:name=*Component*,type=*Component* Config,Application=soa-infra,ApplicationVersion=11.1.1

The *Component* names for the SOA Suite configuration parameters are: `soainfra`, `mediator` and `bpel`.

To use custom WLST commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide* for more information.

10.4 JVM Tuning Parameters

JVM parameters can have an impact on SOA performance. The major factors that impact a SOA component's performance relate to the heap size. For more information on tuning the JVM for performance, see Section 2.4, "Tune Java Virtual Machines (JVMs)".

10.5 Database Settings

Tuning your database configurations may be useful with the SOA Suite of applications. Configurations and specific settings may vary for different use cases. See your database-specific administration manuals for more information on tuning database properties.

For additional basic database tuning guidelines, see Section 2.6, "Tune Database Parameters".

10.5.1 Configuring Data Sources for SOA

SOA obtains database connections using an application server managed data source. You can use the WebLogic Server Console to configure SOA data source. For more information on using the WebLogic Server Console, seethe *Oracle Fusion Middleware Administrator's Guide*.

Consider the following data source configurations when performance is an issue:

- When configuring the data source, ensure that the connection pool has enough free connections.
- Statement caching can eliminate potential performance impacts caused by repeated cursor creation and repeated statement parsing and creation. Statement caching also reduces the performance impact of communication between the application server and the database server
- Disable unnecessary connection testing and profiling.

For more information, see "Tuning JDBC Stores" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

10.5.2 Weblogic Server Performance Tuning

For complete performance tuning of Weblogic Server, refer to *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

11

Oracle BPEL Process Manager Performance Tuning

Oracle Business Process Execution Language (BPEL) Process Manager provides several property settings that can be configured to optimize performance at the composite, fabric, application and server levels. This chapter describes these property settings and provides recommendations on how to use them.

This chapter contains the following sections

- Section 11.1, "About BPEL Process Manager"
- Section 11.2, "Basic Tuning Considerations"
- Section 11.3, "BPEL Properties Set Inside a Composite"
- Section 11.4, "Tables Impacted By Instance Data Growth"

11.1 About BPEL Process Manager

BPEL is the standard for assembling a set of discrete services into an end-to-end process flow, radically reducing the cost and complexity of process integration initiatives. Oracle BPEL Process Manager offers a comprehensive and easy-to-use infrastructure for creating, deploying and managing BPEL business processes.

For more information, see "Configuring BPEL Process Service Components and Engines" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite* and "Using the BPEL Process Service Component" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

11.2 Basic Tuning Considerations

This section describes the basic BPEL Process Manager performance tuning properties that are configured either through WLST or Oracle Enterprise Manager. To modify properties through WLST, see Section 10.3, "Modifying SOA Configuration Parameters".

Note: The configuration examples and recommended settings described in this chapter are for illustrative purposes only. Consult your own use case scenarios to determine which configuration options can provide performance improvements.

11.2.1 BPEL Threading Model

When the dispatcher must schedule a dispatch message for execution, it can enqueue the message into a thread pool. Each dispatch set can contain a thread pool (java.util.concurrent.ThreadPoolExecutor). The BPEL thread pool implementation notifies the threads when a message has been enqueued and ensures the appropriate number of threads are instantiated in the pool.

The following thread properties can be tuned:

- Dispatcher Invoke Threads
- Dispatcher Engine Threads
- Dispatcher System Threads
- Dispatcher Maximum Request Depth

> **Note:** `dspMinThreads`, `dspMaxThreads` and `dspInvokeAllocRatio` configuration properties are deprecated in Oracle 11*g*. In addition, the invoke threads have their own pool in Oracle 11*g* so the `dspInvokeAllocRatio` is no longer required.

11.2.1.1 Dispatcher Invoke Threads

The `dspInvokeThreads` property specifies the total number of threads allocated to process invocation dispatcher messages. Invocation dispatcher messages are generated for each payload received and are meant to instantiate a new instance. If the majority of requests processed by the engine are instance invocations (as opposed to instance callbacks), greater performance may be achieved by increasing the number of invocation threads. Higher thread counts may cause greater CPU utilization due to higher context switching costs.

The minimum number of threads for this thread pool is 1 and it cannot be set to 0 a or negative number.

The default value is 20 threads. Any value less than 1 thread is changed to the default.

11.2.1.2 Dispatcher Engine Threads

The `dspEngineThreads` property specifies the total number of threads allocated to process engine dispatcher messages. Engine dispatcher messages are generated whenever an activity must be processed asynchronously. If the majority of processes deployed are durable with a large number of dehydration points (mid-process receive, onMessage, onAlarm, and wait activities), greater performance may be achieved by increasing the number of engine threads. Note that higher thread counts can cause greater CPU utilization due to higher context switching costs.

The minimum number of threads for this thread pool is 1 and it cannot be set to 0 a or negative number.

The default value is 30 threads. Any value less than 1 thread is changed to the default.

11.2.1.3 Dispatcher System Threads

The `dspSystemThreads` property specifies the total number of threads allocated to process system dispatcher messages. System dispatcher messages are general clean-up tasks that are typically processed quickly by the server (for example, releasing stateful message beans back to the pool). Typically, only a small number of threads are required to handle the number of system dispatch messages generated during run time.

The minimum number of threads for this thread pool is 1 and it cannot be set to 0 a or negative number.

The default value is `2`. Any value less than 1 thread is changed to the default.

11.2.1.4 Dispatcher Maximum Request Depth

The `dspMaxRequestDepth` property sets the maximum number of in-memory activities to process within the same request. After processing an activity request, Oracle BPEL Process Manager attempts to process as many subsequent activities as possible without jeopardizing the validity of the request. Once the activity processing chain has reached this depth, the instance is dehydrated and the next activity is performed in a separate transaction.

If the request depth is too large, the total request time can exceed the application server transaction time out limit.This process is applicable to durable processes.

The default value is 600 activities.

> **Note:** Note that the minimum number of threads for each thread pool is 1. dsp*Threads can not be set to 0 or negative.

11.2.2 Audit Level

The `auditLevel` property sets the audit trail logging level. This configuration property is applicable to both durable and transient processes. This property controls the amount of audit events that are logged by a process. Audit events result in more database inserts into the `audit_trail` table which may impact performance. Audit information is used only for viewing the state of the process from Oracle Enterprise Manager Console.

Use the Off value if you do not want to store any audit information. Always choose the audit level according to your business requirements and use cases. For more information on setting the audit level, see "Understanding the Order of Precedence for Audit Level Settings" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

Value	Description
Inherit	Inherits the audit level from infrastructure level.
Off	No audit events (activity execution information) are persisted and no logging is performed; this can result in a slight performance boost for processing instances.
Minimal	All events are logged; however, no audit details (variable content) are logged.
Production	All events are logged. The audit details for assign activities are not logged; the details for all other activities are logged.
Development	All events are logged; all audit details for all activities are logged.

11.2.3 OneWayDeliveryPolicy

The `oneWayDeliveryPolicy` is from the Oracle 10*g* configuration property `deliveryPersistencePolicy`.

The new configuration property name is `bpel.config.oneWayDeliveryPolicy`.

WARNING: If you set this property to `async.cache` and your system fails, you may lose messages. For more information, refer to the *Oracle BPEL Process Manager Administrator's Guide*.

The `oneWayDeliveryPolicy` property controls database persistence of messages entering Oracle BPEL Server. By default, incoming requests are saved in the delivery service database table `dlv_message`. These requests are later acquired by Oracle BPEL Server worker threads and delivered to the targeted BPEL process. This property persists delivery messages and is applicable to durable processes.

If you set the `oneWayDeliveryPolicy` property to `async.cache` and your system fails, you may lose messages. In addition, the system can become overloaded (messages become backlogged in the scheduled queue) and you may receive out-of-memory errors. Consult your own use case scenarios to determine if this setting is appropriate.

One-way invocation messages are stored in the delivery cache until delivered. If the rate at which one-way messages arrive is much higher than the rate at which Oracle BPEL Server delivers them, or if the server fails, messages may be lost.

Value	Description
async.persist (Default)	Delivery messages are persisted in the database. With this setting, reliability is obtained with some performance impact on the database. In some cases, overall system performance can be impacted.
async.cache	Incoming delivery messages are kept only in the in-memory cache. If performance is preferred over reliability, this setting should be considered.
sync	Directs Oracle BPEL Server to bypass the scheduling of messages in the invoke queue, and invokes the BPEL instance synchronously. In some cases this setting can improve database performance.

11.2.4 StatsLastN

The `StatsLastN` property sets the size of the most-recently processed request list. After each request is finished, statistics for the request are kept in a request list. A value less than or equal to 0 disables statistics gathering. To optimize performance, consider disabling statistics collection if you do not need them.

This property is applicable to both durable and transient processes.

The default value is -1.

11.2.5 AuditDetailThreshold

The `auditdetailthreshold` property sets the maximum size (in kilobytes) of an audit trail details string before it is stored separately from the audit trail. If an audit trail details string is larger than the threshold setting, it is not immediately loaded when the audit trail is initially retrieved; a link is displayed with the size of the details string. Strings larger than the threshold setting are stored in the `audit_details` table, instead of the `audit_trail` table.

The details string typically contains the contents of a BPEL variable. In cases where the variable is very large, performance can be severely impacted by logging it to the audit trail.

The default value is 50000 (50 kilobytes).

11.2.6 LargeDocumentThreshold

The `largedocumentthreshold` property sets the large XML document persistence threshold. This is the maximum size (in kilobytes) of a BPEL variable before it is stored in a separate location from the rest of the instance scope data.

This property is applicable to both durable and transient processes.

Large XML documents impact the performance of the entire Oracle BPEL Server if they are constantly read in and written out whenever processing on an instance must be performed.

The default value is 10000 (100 kilobytes).

11.2.7 Validate XML

The `validateXML` property validates incoming and outgoing XML documents. If set to True, the Oracle BPEL Process Manager applies schema validation for incoming and outgoing XML documents.

This property is applicable to both durable and transient processes.

The default value is False.

11.2.8 SyncMaxWaitTime

The `SyncMaxWaitTime` property sets the maximum time the process result receiver waits for a result before returning. Results from asynchronous BPEL processes are retrieved synchronously by a receiver that waits for a result from Oracle BPEL Server.

This property is applicable to transient processes.

The default value is 45 seconds.

11.2.9 InstanceKeyBlockSize

The `InstanceKeyBlockSize` property controls the instance ID range size. Oracle BPEL Server creates instance keys (a range of process instance IDs) in batches using the value specified. After creating this range of in-memory IDs, the next range is updated and saved in the `ci_id_range` table.

For example, if `instanceKeyBlockSize` is set to `100`, Oracle BPEL Server creates a range of instance keys in-memory (100 keys, which are later inserted into the `cube_instance` table as `cikey`). To maintain optimal performance, ensure that the block size is larger than the number of updates to the `ci_id_range` table.

The default value is 10000.

11.3 BPEL Properties Set Inside a Composite

This section lists the config properties of some sections of the deployment descriptor. For each configuration property parameter, a description is given, as well as the expected behavior of the engine when it is changed.

All the properties set in this section affect the behavior of the component containing the BPEL process only. Each BPEL process can be created as a component of a composite. These properties are modified through WLST.

11.3.1 Component Properties

The following Component properties can be tuned for performance:

11.3.1.1 inMemoryOptimization

This property indicates to Oracle BPEL Server that this process is a transient process and dehydration of the instance is not required. When set to True, the completionPersistPolicy is used to determine persistence behavior. This property can only be set to True for transient processes or processes that do not contain any dehydration points such as receive, wait, onMessage and onAlarm activities. The inMemoryOptimization property is set at the BPEL component level.

Values:

This property has the following values:

- False (default): instances are persisted completely and recorded in the dehydration store database.
- True: The completionPersist policy is used to determine persistence behavior. See Section 11.3.1.2.

11.3.1.2 completionPersistPolicy

This property configures how the instance data is saved. It can only be set at the BPEL component level. The completionPersistPolicy property can only be used when inMemoryOptimization is set to be True (transient processes). Note that this parameter may affect database growth and throughput (due to reduced I/O).

Value	Description
On (default)	The completed instance is saved normally
Deferred	The completed instance is saved, but with a different thread and in another transaction.
Faulted	Only the faulted instances are saved.
Off	No instances of this process are saved.

11.3.2 Partner Link Property

You can dynamically configure a partner link at runtime in BPEL. This is useful for scenarios in which the target service that BPEL wants to invoke is not known until runtime. The following Partner Link properties can be tuned for performance:

11.3.2.1 idempotent

An idempotent activity is an activity that can be retried (for example, an assign activity or an invoke activity). Oracle BPEL Server saves the instance after a nonidempotent activity. This property is applicable to both durable and transient processes.

Values:

This property has the following values:

- False: Activity is dehydrated immediately after execution and recorded in the dehydration store. When `idempotent` is set to False, it provides better failover protection, but may impact performance if the BPEL process accesses the dehydration store frequently.

- True (default): If Oracle BPEL Server fails, it performs the activity again after restarting. This is because the server does not dehydrate immediately after the invoke and no record exists that the activity executed. Some examples of where this property can be set to True are: read-only services (for example, CreditRatingService) or local EJB/WSIF invocations that share the instance's transaction.

11.3.2.2 nonBlockingInvoke

By default, Oracle BPEL Process Manager executes in a single thread, executing the branches sequentially instead of in parallel. When this property is set to True, the process manager creates a new thread to perform each branch's invoke activity in parallel. This property is applicable to both durable and transient processes.

Consider setting this property to True if you have invoke activities in multiple flow or flow *n* branches. This is especially effective if the parallel invoke activities are two-way, but some benefits can be realized for parallel one-way invokes as well.

Values:

This property has the following values:

- True: Oracle BPEL Server spawns a new thread to execute the invocation.
- False (default): Oracle BPEL Server executes the invoke activity in the single process thread.

11.3.2.3 validateXML

Enables message boundary validation. Note that additional validation can impact performance by consuming extra CPU and memory resources.

Values:

- True: When set to True the engine validates the XML message against the XML schema during <receive> and <invoke> for this partner link. If the XML message is invalid then `bpelx:invalidVariables` run time BPEL Fault is thrown. This overrides the domain level `validateXML` property.
- False (default): Disables XML validation.

11.4 Tables Impacted By Instance Data Growth

Instance data occupies space in Oracle BPEL Process Manager schema tables. Data growth from auditing and dehydration can have a significant impact on database performance and throughput. See Section 11.2.2, "Audit Level" for audit configuration and Section 11.3.1.1, "inMemoryOptimization" for dehydration configuration. The table below describes the tables that are impacted by instance data growth. A brief description is provided of each table.

Table 11–1 Oracle BPEL Process Manager Tables Impacted by Instance Data Growth

Table Name	Table Description
`audit_trail`	Stores the audit trail for instances. The audit trail viewed in Oracle BPEL Control is created from an XML document. As an instance is processed, each activity writes events to the audit trail as XML.

Table 11–1 (Cont.) Oracle BPEL Process Manager Tables Impacted by Instance Data

Table Name	Table Description
audit_details	Stores audit details that can be logged through the API. Activities such as an assign activity log the variables as audit details by default. Audit details are separated from the audit_trail table due to their large size. If the size of a detail is larger than the value specified for this property, it is placed in this table. Otherwise, it is placed in the audit_trail table.
cube_instance	Stores process instance metadata (for example, the instance creation date, current state, title, and process identifier)
cube_scope	Stores the scope data for an instance (for example, all variables declared in the BPEL flow and some internal objects that help route logic throughout the flow).
dlv_message	Stores incoming (invocation) and callback messages upon receipt. This table only stores the metadata for a message (for example, current state, process identifier, and receive date).
dlv_subscription	Stores delivery subscriptions for an instance. Whenever an instance expects a message from a partner (for example, the receive or onMessage activity) a subscription is written out for that specific receive activity.
document_ci_ref	Stores cube instance references to data stored in the xml_document table.
document_dlv_msg_ref	Stores references to dlv_message documents stored in the xml_document table.
schema_md	Stores metadata about columns defined in the Oracle BPEL Process Manager schema (orabpel).
task	Stores tasks created for an instance. The TaskManager process keeps its current state in this table.
work_item	Stores activities created by an instance. All activities in a BPEL flow have a work_item table. This table includes the metadata for the activity (current state, label, and expiration date (used by wait activities)).
xml_document	Stores all large objects in the system (for example, dlv_message documents). This table stores the data as binary large objects (BLOBs). Separating the document storage from the metadata enables the metadata to change frequently without being impacted by the size of the documents.
Header_properties	stores headers and properties information

12

Oracle Mediator Performance Tuning

This chapter describes how to tune Oracle Mediator for optimal performance. It contains the following topics:

- Section 12.1, "About Oracle Mediator"
- Section 12.2, "Basic Tuning Considerations"
- Section 12.3, "Event Delivery Network (EDN) Tuning"

12.1 About Oracle Mediator

Mediator is a component of Oracle SOA offering that provides mediation capabilities like selective routing, transformation and validation capabilities, along with various message exchange patterns, like synchronous, asynchronous and event publishing or subscription. Oracle Mediator provides the framework to mediate between various providers and consumers of services and events. The Mediator service engine runs with the SOA Service Infrastructure Java EE application.

See Also: For details about the SOA Suite, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

For details about Oracle Mediator, see "Administering Oracle Mediator Service Components and Engines" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

12.2 Basic Tuning Considerations

In most business environments, customer data resides in disparate sources including business partners, legacy applications, enterprise applications, databases, and custom applications. The challenge of integrating this data efficiently can be met by using Oracle Mediator to deliver real-time data access to all applications that update or have a common interest in the same data.

Note: Before you begin tuning Oracle Mediator properties, be sure that you have read and understand the Oracle Mediator chapters in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*.

This section provides details about setting common Oracle Mediator properties such as:

- metricsLevel

- Domain-Value Maps
- Deferred Routing Rules
- Error and Retry Parameters

12.2.1 metricsLevel

This property controls DMS metrics tracking level. By default, DMS metrics collections is enabled. If you do not need to collect DMA metrics data, consider setting the `metricsLevel` to Disabled to improve performance.

12.2.2 Domain-Value Maps

When performance is an issue, consider using domain-value maps instead of database lookup within XSL transformations to minimize file I/O.

For more information on using domain value maps, see "Working with Domain Value Maps" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

12.2.3 Deferred Routing Rules

The following performance configuration parameters can be used for tuning components with parallel routing rules deployed:

- `DeferredWorkerThreadCount`: Specifies the number of deferred dispatchers for processing messages in parallel. For higher loads consider increasing this parameter to have more number of outbound threads for deferred processing as each parallel rule is processed by one of the `DeferredWorkerThreads`. Default value is 4 threads.
- `DeferredMaxRowsRetrieved`: When Mediator routing rule type is set to 'Parallel', `DeferredMaxRowsRetrieved` sets the number of maximum rows (maximum number of messages for parallel routing rule processing) that are retrieved from Mediator store table (that stores messages for parallel routing rule) for processing. Note that each message retrieved in this batch is processed by one worker thread at a time. Default value is 20 threads.
- `DeferredLockerThreadSleep`: For processing parallel routing rules, Oracle Mediator has a daemon locker thread that retrieves and locks messages from Mediator store database. The thread polls the database to retrieve messages for parallel processing. When no messages are available, the locker thread "sleeps" for the amount of time specified in the `DeferredLockerThreadSleep` and prevents round trips to database. Default value is 2 seconds.

  During the specified time, no messages are available for parallel routing in either of the following cases:

  – There are no Mediator components with parallel routing rules deployed.

  – Mediator component(s) with parallel routing rule is deployed, but there are no continuous incoming messages for such components.

  Note: You can specify Oracle Mediator component Priority through JDeveloper Mediator designer. This property is used to set priority among Oracle Mediator components with parallel routing rules.

  For more information, see "Creating Mediator Routing Rules" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

12.2.4 Error and Retry Parameters

Consider increasing the following error and retry parameter values when you do not want to reduce the number of database trips:

- `ErrorLockerThreadSleep`: Specifies the idle time between two successive iterations for retrieving errored out messages, when there is no errored out message from parallel processing. The time is measured in seconds. Default value is 5 seconds.
- `RetryLockerThreadSleep`: Sleep time in seconds for retry locker when there is no message. Default value is 5 seconds.

For more information on routing, see "Creating Mediator Routing Rules" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

12.3 Event Delivery Network (EDN) Tuning

The Event Delivery Network (EDN) delivers events published by Oracle Mediator, Oracle BPEL Process Manager components, and external publishers such as Oracle Application Development Framework entity objects.

To improve performance of the Event Delivery Network, consider increasing the thread count (default is 3.) This property can be modified through WLST. For more information, see Section 10.3, "Modifying SOA Configuration Parameters".

13

Oracle Human Workflow Performance Tuning

This chapter describes how to tune Oracle Human Workflow for optimal performance. You can tune Oracle Human Workflow in these areas:

- Section 13.1, "About Oracle Human Workflow"
- Section 13.2, "Human Workflow Tuning Considerations"
- Section 13.3, "Improving Server Performance"
- Section 13.4, "Completing Workflows Faster"
- Section 13.5, "Tuning Identity Provider"
- Section 13.6, "Tuning the Database"

13.1 About Oracle Human Workflow

Oracle Human Workflow is a service engine running in Oracle SOA Service Infrastructure that allows the execution of interactive human driven processes. A human workflow provides the human interaction support such as approve, reject, and reassign actions within a process or outside of any process. The Human Workflow service consists of a number of services that handle various aspects of human interaction with a business process.

For more information, see "Using the Human Workflow Service Component" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

See also the Oracle Human Workflow web site at `http://www.oracle.com/technology/products/soa/hw/index.html`.

13.2 Human Workflow Tuning Considerations

This section discusses the various options available to address these factors:

- Minimize Client Response Time
- Choose the Right Workflow Service Client
- Narrow Qualifying Tasks Using Precise Filters
- Retrieve Subset of Qualifying Tasks (Paging)
- Fetch Only the Information That Is Needed for a Qualifying Task
- Reduce the Number of Return Query Columns
- Use the Aggregate API for Charting Task Statistics

- Use the Count API Methods for Counting the Number of Tasks
- Create Indexes On Demand for Flexfields
- Use the doesTaskExist Method

13.2.1 Minimize Client Response Time

Since workflow client applications are interactive, it is important to have good response time at the client. Some of the factors that affect the response time include service call performance impacts, querying time to determine the set of qualifying tasks for the request, and the amount of additional information to be retrieved for each qualifying task.

13.2.2 Choose the Right Workflow Service Client

Workflow services support two major types of clients: SOAP and EJB clients. EJB clients can be further separated into local EJB clients and remote EJB clients.

If the client application is based on .Net technologies, then only the SOAP workflow services can be used. However, if the client application is based on J2EE technology, then consider which client should be used based on your use case scenarios. The options are listed below:

- Remote client - This is the best option in terms of performance in most cases. If the client is running in the same JVM as the workflow services (soa-infra application), the API calls are optimized so that there is no remote method invocation (RMI) involved. If the client is on a different JVM, then RMI is used, which can impact performance due to the serialization and de-serialization of data between the API methods.
- SOAP client - While this option is preferred for standardization (based on web services), there are additional performance considerations when compared to the remote method invocation (RMI) used in the remote client. Additional processing is performed by the web-services technology stack which causes the marshalling and unmarshalling of API method arguments between XML.

For more information, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

13.2.3 Narrow Qualifying Tasks Using Precise Filters

Using precise filters is one of the most important factors in improving response time. When a task list is retrieved, the query should be as precise as possible so the maximum filtering can be done at the database level.

For example, when the inbox view is requested for a user, the tasks are filtered mainly based on whether they are assigned to the current user or to the groups the user belongs to. By specifying additional predicate filters on the inbox view, the overall response time for the query can be reduced since lesser number of tasks qualify.

Alternatively, you can define views by specifying predicate filters and the overall response time for such views is reduced since lesser number of tasks qualify. All predicates passed to the query APIs (or defined in the views) are directly pushed to the database level SQL queries. With this information, the database optimizer can use the best indexes to create an optimal execution plan. The additional filters can be based on task attributes or promoted flex fields. For example, instead of listing all PO approval tasks, views can be defined to present tasks to the user based on priority, date, category, or amount range.

Example: To retrieve all assigned tasks for a user with priority = 1, you can use the following API call:

```
Predicate pred = new Predicate(TableConstants.WFTASK_STATE_COLUMN,
Predicate.OP_EQ,
IWorkflowConstants.TASK_STATE_ASSIGNED);
pred.addClause(Predicate.AND,
TableConstants.WFTASK_PRIORITY_COLUMN,
Predicate.OP_EQ,

1);
List tasks = querySvc.queryTasks(ctx,
queryColumns,
null,
ITaskQueryService.AssignmentFilter.MY ITaskQueryService.AssignmentFilter.MY,
null,
pred,
null,
startRow,
endRow);
```

13.2.4 Retrieve Subset of Qualifying Tasks (Paging)

Once the task list has been narrowed down to meet a specific criteria as discussed in the previous section, the next level of filtering is based on how many tasks are to be presented to the user. You want to avoid fetching too many rows, which not only increases the query time but also increases the application process time and the amount of data returned to client. The query API has paging parameters that control the number of qualifying rows returned to the user and the start row.

For example, in the `queryTasks` method:

```
List tasks = querySvc.queryTasks(ctx,
queryColumns,
null,
 ITaskQueryService.AssignmentFilter.MY,
null,
pred,
null,
startRow,
endRow);
```

Consider setting the `startRow` and `endRow` parameters to values that may limit the number of return matching records.

13.2.5 Fetch Only the Information That Is Needed for a Qualifying Task

When using the queryTask service, consider reducing the amount of optional information retrieved for each task returned in the list. This may reduce the performance impacts from additional SQL query and Java logic.

For example, in the following `queryTasks` method, only the group actions information is retrieved. You can also retrieve attachment and payload information directly in the listing, but you may encounter performance impacts.

```
List<ITaskQueryService.OptionalInfo> optionalInfo
= new ArrayList<ITaskQueryService.OptionalInfo>();
optionalInfo.add(ITaskQueryService.OptionalInfo.GROUP_ACTIONS);
// optionalInfo.add(ITaskQueryService.OptionalInfo.ATTACHMENTS);
// optionalInfo.add(ITaskQueryService.OptionalInfo.PAYLOAD);
List tasks = querySvc.queryTasks(ctx,
queryColumns,
```

```
optionalInfo,
ITaskQueryService.AssignmentFilter.MY,
null,
pred,
null,
startRow,
endRow);
```

In rare cases where the entire payload is needed, then the payload information can be requested. Typically only some of the payload fields are needed for displaying the task list. For example, for PO Tasks, the PO amount may be a column that must be displayed. Rather than fetching the payload as additional information and then retrieving the amount using an xpath expression and displaying it in the listing, consider mapping the amount column from the payload to a flex field. The flex field can then be directly retrieved during SQL querying which may significantly reduce the processing time.

Similarly, for attachments where the name of the attachment is to be displayed in the listing and the document itself is stored in an external repository, consider capturing the attachment name in the payload and mapping it to a flex field, so that processing time is optimized. While constructing the listing information, the link to the attachment can be constructed by fetching the appropriate flex field.

13.2.6 Reduce the Number of Return Query Columns

When using the `queryTask` service, consider reducing the number of query columns to improve the SQL time. Also, try to use the common columns as they are most likely indexed and the SQL can execute faster.

For example, in the following `queryTasks` method, only the `TASKNUMBER` and `TITLE` columns are returned:

```
List queryColumns = new ArrayList();
queryColumns.add("TASKNUMBER");
queryColumns.add("TITLE");
...
List tasks = querySvc.queryTasks(ctx,
null,
 ITaskQueryService.AssignmentFilter.MY,
null,
pred,
null,
startRow,
endRow);
```

13.2.7 Use the Aggregate API for Charting Task Statistics

Sometimes it is necessary to display charts or statistics to summarize task information. Rather than fetching all the tasks using the query API, and computing the statistics at the client layer, consider using the new aggregate APIs to compute the statistics at the database level.

For example, the following call illustrates the use of the API to get summarized statistics based on state for tasks assigned to a user:

```
List taskCounts = querySvc.queryAggregatedTasks(ctx,
Column.getColumn(WFTaskConstants.STATE_COLUMN),
 ITaskQueryService.AssignmentFilter.MY,
keyWordFilter,
filterPredicate,
false,
```

```
false);
```

13.2.8 Use the Count API Methods for Counting the Number of Tasks

Sometimes it is only necessary to count how many tasks exist that match certain criteria. Rather than calling the `queryTasks` API method, and determining the size of the returned list, call the `countTasks` API method, which returns only the number of matching tasks. The performance impact of returning a count of tasks is much lower than returning a list of task objects.

For example, the following call illustrates the use of the API to get the total number of tasks assigned to a user:

```
int numberOfTasks = querySvc.countTasks(ctx,
ITaskQueryService.AssignmentFilter.MY,
keyWordFilter,
filterPredicate);
```

13.2.9 Create Indexes On Demand for Flexfields

The workflow schema table WFTASK contains several flexfield attribute columns that can be used for storing task payload values in the workflow schema. Because there are numerous columns, and their use is optional, the installed schema does not contain indexes for these columns. In certain use-cases, for example, where certain mapped flexfield columns are frequently used in query predicates, performance can be improved if you create indexes on these columns.

For example, to create an index on the TEXTATTRIBUTE1 column, the following SQL command should be run:

```
create index WFTASKTEXTATTRIBUTE1_I on WFTASK(TEXTATTRIBUTE1);
```

Note: The exact indexes required depend on the flexfield attribute columns being used, and the nature of the queries being executed. After creating the indexes, the statistics for the WFTASK table should be re-computed and flushed.

13.2.10 Use the doesTaskExist Method

Sometimes it is necessary to check whether any tasks exist that match particular query criteria. Rather than calling the `countTasks` method, and checking if the number returned is zero, consider using `doesTaskExist`. The `doesTaskExist` method performs an optimized query that simply checks if any rows exist that match the specified criteria. This method may achieve better results than calling the `countTasks` method.

For example, the following call illustrates the use of the API method to determine if a user owns any task instances:

```
boolean userOwnsTask = querySvc.doesTaskExist(ctx,
ITaskQueryService.AssignmentFilter.OWNER,null,null);
```

13.3 Improving Server Performance

Server performance essentially determines the scalability of the system under heavily loaded conditions. Section 13.2.1, "Minimize Client Response Time" lists several ways in which client response times can be minimized by fetching the right of amount of

information and reducing the potential performance impact associated with querying. These techniques also reduce the database and service logic performance impacts at the server and can improve server performance. In addition, a few other configuration changes can be made to improve server performance:

- Archive Completed Instances Periodically
- Select the Appropriate Workflow Callback Functionality
- Minimize Performance Impacts from Notification
- Deploy Clustered Nodes

13.3.1 Archive Completed Instances Periodically

The database scalability of a system is largely dependent on the amount of data in the system. Since business processes and workflows are temporal in nature, once they are processed, they are not queried frequently. Having numerous completed instances in the system can slow the system. Consider using an archival scheme to periodically move completed instances to another system that can be used to query historical data. Archival should be done carefully to avoid orphan task instances.

13.3.2 Select the Appropriate Workflow Callback Functionality

The workflow callback functionality can be used to query or update external systems after any significant workflow event, such as assignment or completion of task. While this functionality is very useful, it has to be implemented correctly to avoid impacting performance.

When performance is critical, ensure that there are sufficient resources to update the external system after the task is completed instead of after every workflow event. For example, instead of using a callback, the service can be invoked once after the completion of the task. If a callback cannot be avoided, then consider using a Java callback instead of a BPEL callback. Java callbacks do not have the performance impact associated with a BPEL callback since the callback method is executed in the same thread. In contrast, a BPEL callback may impact performance when sending a message to the BPEL engine, which in turn must be correlated so that it is delivered to the correct process instance. The workflow service has to be called by the BPEL engine after the invocation of the service.

13.3.3 Minimize Performance Impacts from Notification

Notifications are useful for alerting users that they have a task to execute. In environments where most approvals happen through email, actionable notifications are especially useful. This also implies that there is not much load in terms of worklist usage. However if most users interact through the Worklist, and notifications serve a secondary purpose, then notifications should be used judiciously. Consider minimizing the notification to just alert a user when a task is assigned instead of sending out notifications for each workflow event. Also, if the task content is also mailed in the notification there may be an impact to performance. To minimize the impact, consider making the notifications secure in which case only a link to the task is sent in the notification and not the task content itself.

13.3.4 Deploy Clustered Nodes

All workflow instances and state information are stored in the dehydration database. Workflow services are stateless which means they can be used concurrently on a cluster of nodes. When performance is critical and a highly scalable system is needed,

a clustered environment can be used for supporting workflow. For more information on clustered architecture, see Section 23.2, "Using Clusters with Oracle Fusion Middleware".

13.4 Completing Workflows Faster

The time it takes for a workflow to complete depends on the routing type specified for the workflow. The workflow functionality provides some options that can be used to improve the amount of time it takes to complete workflows. Some of these options are discussed in this section:

- Use Workflow Reports to Monitor Progress
- Specify Escalation Rules
- Specify User and Group Rules for Automated Assignment
- Use Task Views to Prioritize Work

13.4.1 Use Workflow Reports to Monitor Progress

Several workflow reports (and corresponding views) are available that can make monitoring and proactively fixing problems easier. A few of these reports are listed below:

- The Unattended Tasks Report provides a list of group tasks that need attention since they have not yet been acquired by any user to work on.
- The Task Cycle Time Report gives an idea of how much time it takes for a particular type of workflow to complete.
- The Task Productivity Report indicates the inflow and outflow of tasks for different users.
- The Assignee Time Distribution Report provides a detailed drill-down of the time spent by each user during the task life cycle (including the idle time when the task was waiting to be picked up by a user.)

All of these reports can be used effectively to fix problems. By checking unattended tasks report, you can assign tasks that have been in the queue for a long time to specific users. By monitoring cycle time and other statistics, you can add staff to groups that are overloaded or take a longer time to complete. Thus reports can be used effectively to ensure workflows complete faster.

13.4.2 Specify Escalation Rules

To ensure that tasks do not get stuck at any user, you can specify escalation rules. For example, you can move a task to a manager if a certain amount of time passes without any action being taken on the task. Custom escalation rules can also be plugged in if the task must be escalated to some other user based on alternative routing logic. By specifying proper escalation rules, you can reduce workflow completion times.

13.4.3 Specify User and Group Rules for Automated Assignment

Instead of manually reassigning tasks to other users or members of a group, you can use user and group rules to perform automated reassignment. This ensures that workflows get timely attention. For example, a user can set up a user rule such that workflows of a specific type and matching a certain filter criteria are automatically reassigned to another user in a specified time window. Similarly, a group rule can be

used to automatically reassign workflows to a member of the group based on different routing criteria such as round robin or most productive. Thus rules can help significantly reduce workflow waiting time, which results in faster workflow completion.

13.4.4 Use Task Views to Prioritize Work

A user's inbox can contain tasks of various types with various due dates. The user has to manually sift through the tasks or sort them to find out which one he or she should work on next. Instead, by creating task views where tasks are filtered based on due dates or priority, users can get their work prioritized automatically so they can focus on completing their tasks instead of wasting their time on deciding which tasks to work on. This also results in faster completion of workflows.

13.5 Tuning Identity Provider

The workflow service uses information from the identity provider in constructing the SQL query to determine the tasks qualifying for a user based on his or her role/group membership. The identity provider is also queried for determining role information to determine privileges of a user when fetching the details of a task and determining what actions can the user perform on a task. There are a few ways to speed up requests made to the identity provider.

- Set the search base in the identity configuration file to node(s) as specific as possible. Ideally you should populate workflow-related groups under a single node to minimize traversal for search and lookup. This is not always possible; for example, you may need to use existing groups and grant membership to groups located in other nodes. If it is possible to specify filters that can narrow down the nodes to be searched, then you should specify them in the identity configuration file.
- Index all critical attributes such as dn and cn in the identity provider. This ensures that when a search or a lookup is done, only a subset of the nodes are traversed instead of a full tree traversal.
- Use an identity provider that supports caching. Not all LDAP providers support caching but Oracle Internet Directory supports caching which can make lookup and search queries faster.

13.6 Tuning the Database

The Human Workflow schema is shipped with several indexes defined on the most important columns for all the tables. Based on the type of request, different SQL queries are generated to fetch the task list for a user. The database optimizer evaluates the cost of different plan alternatives (for example, full table scan, access table by index) and decides on a plan that is lower in cost. For the optimizer to work correctly, the index statistics should be current at all times. As with any database usage, it is important to make sure the database statistics are updated at regular intervals and other tunable parameters such as memory, table space, and partitions are used effectively to get maximum performance.

For more information on tuning the database, see Section 2.6, "Tune Database Parameters".

14

Oracle Adapters Performance Tuning

This chapter describes how to tune Oracle Adapters for optimal performance. Oracle Adapters, a component of the Oracle SOA Suite of Applications, provide an integrated view of data and allow multiple applications to be integrated.

This chapter contains the following sections:

- Section 14.1, "About Oracle Adapters"
- Section 14.2, "Oracle JCA Adapters for Files/FTP"
- Section 14.3, "Oracle JCA Adapter for Database Tuning"
- Section 14.4, "Oracle Socket Adapter Tuning"
- Section 14.5, "Oracle SOA JMS Adapter Tuning"
- Section 14.6, "Oracle AQ Adapter Tuning"
- Section 14.7, "Oracle MQ Adapter Tuning"

14.1 About Oracle Adapters

Oracle technology adapters integrate Oracle Application Server and Oracle Fusion Middleware components such as Oracle BPEL Process Manager (Oracle BPEL PM) or Oracle Mediator components to file systems, FTP servers, database queues (advanced queues, or AQ), Java Message Services (JMS), database tables, and message queues (MQ Series).

For more information on Oracle Adapters, see *Oracle Fusion Middleware User's Guide for Technology Adapters*.

14.2 Oracle JCA Adapters for Files/FTP

This section describes the various features available for scalability and performance tuning of Oracle File and FTP Adapters.The Oracle File and FTP Adapters provide knobs to throttle the inbound and outbound operations. The Oracle File and FTP Adapters also provide knobs that can be used to tune the performance of outbound operations. The Oracle File and FTP Adapters knobs are described in the following sections:

- Inbound Throttling Best Practices
- Outbound Throttling Best Practices
- Outbound Performance Best Practices

Note: For composites with Oracle File and FTP Adapters, which are designed to consume very large number of concurrent messages, you must set the number of open files parameter for your operating system to a larger value. For example, to set the number of open files parameter to 8192 for Linux, use the `ulimit -n` 8192 command

14.2.1 Inbound Throttling Best Practices

The Oracle File and FTP Adapters provide parameters that can be used to throttle the inbound operations. The table below describes the inbound throttling practices:

Parameter	Type	Values	Description
`MaxRaiseSize`	JCA	`<property name="MaxRaiseSize" value="100"/>` Default: 10000 (ten thousand)	This parameter defines the maximum number of files that the inbound adapter would submit for processing on each polling cycle. For example, if your inbound directory has 1000 files and the `MaxRaiseSize` is set to `100`, the adapter can increase to 100 files on each polling cycle. Defined in the Inbound JCA File.
`SingleThreadModel`	JCA	`<property name="SingleThreadModel" value="true"/>` Default: False (In this case, the global in-memory queue is used).	If the value is `true`, the poller lists, translates, or publishes files in the same thread. In other words, it does not use the global in-memory queue for publishing. Defined in the Inbound JCA File.
`ThreadCount`	JCA	`<property name="ThreadCount" value="10"/>` Default: -1 (In this case, the adapter uses the global thread pool and in-memory queue)	This parameter enables the Oracle File and FTP Adapters to create their own processor threads rather than depending on the global pool of processor worker threads for processing the enqueued files. This parameter partitions the in-memory queue and each composite application receives its own in-memory queue. If the `ThreadCount` is set to `0`, then the threading behavior is the same as that of the SingleThreadModel. If the `ThreadCount` is set to `-1`, then the global thread pool is activated, which is the same as the Default Threading Model. The maximum value that can be set for `ThreadCount` is `40`. Defined in the Inbound JCA File.

14.2.2 Outbound Throttling Best Practices

The Oracle File and FTP Adapters provide parameters that can be used to throttle the outbound operations. The table below describes the outbound throttling practices:

Parameter	Type	Value	Description
ConcurrentThreshold	JCA	<property name="ConcurrentThreshold" value="100"/> Default: 20 (In this case, not more than 20 translations occur for a particular outbound scenario.)	This parameter specifies the maximum number of translation activities that are allowed to start in parallel for a particular outbound scenario. The translation step during the outbound operation is CPU intensive and must be monitored as it might cause other applications or threads to starve. The maximum value is 100. Defined in the Outbound JCA File.

14.2.3 Outbound Performance Best Practices

The Oracle File and FTP Adapters provide parameters that can be used to tune the performance of outbound operations. The table below describes the outbound performance parameters:

Parameter	Type	Value	Description
UseStaging	JCA	<property name="UseStaging" value="true"/> Default: True	If the parameter is set to true, then the outbound Oracle File or FTP Adapter writes translated data to a staging file and later streams the staging file to the target file. If the parameter is set to false, then the outbound Oracle File or FTP Adapter does not use an intermediate staging file. Defined in Outbound JCA File.
serializeTranslation	Endpoint Property	<reference name="PurchaseOrderOut"> <interface.wsdl interface="...."/> <binding.jca config="PurchaseOrderOut_ftp.jca"/> <property name="serializeTranslation" type="xs:string" many="false" source="" override="may">true</property> </reference> Defaults: ■ True (If the value of UseStaging is set to True) ■ False (If the value of UseStaging is set to False)	If True, then the translation step is serialized using a semaphore. The number of permits for semaphore (monitoring the translation step) comes from ConcurrentThreshold parameter (listed in the preceding table). The default value of True is used because the translation step is CPU intensive and you do not want to starve other applications or threads. If False, then the translation step occurs outside the semaphore. Defined in Binding property for reference in composite.xml.

Parameter	Type	Value	Description
inMemoryTranslation	Binding Property	<reference name="PurchaseOrderOut"> <interface.wsdl interface="...."/> <binding.jca config="PurchaseOrderOut_ftp.jca"/> <property name="inMemoryTranslation" type="xs:string" many="false" source=""override="may">false</property> </reference> Default: False	This parameter is applicable only if UseStaging is False. If True, then the translation step occurs in-memory (an in-memory byte array is created.) If False, then the adapter creates an output stream to the target file (FTP, FTPS, and SFTP included) and allows the translator to translate and write directly to the stream. Defined in Binding property for reference in composite.xml.

14.3 Oracle JCA Adapter for Database Tuning

The Oracle Database Adapter is pre-configured with many performance optimizations. You can, however, make some changes to reduce the number of round trips to the database, as described in the following sections:

- JCA Adapter Basic Tuning Considerations
- Existence Checking

> **Note:** The tuning considerations in this chapter are listed for example only. Tuning parameters are specific to each deployment. Review you current usage and performance issues to determine which tuning considerations can improve performance.

14.3.1 JCA Adapter Basic Tuning Considerations

Adapter performance is directly related to the number of round-trips to the database, and the network cost of each trip. If performance becomes an issue, and making modifications is appropriate for your deployment, consider tuning the following parameters:

- Use Indexes

  Indexes can improve performance of selects, updates and deletes. Index all queried fields, such as the primary key and the MarkReadField of the LogicalDeletePollingStrategy, when polling. For MarkReadField specify a non-null MarkUnreadValue. Caution: An index on a column containing many nulls may revert to full table scans.

- Disable OptimizeMerge

  The OptimizeMerge parameter allows the detection of XML elements for which no value was specified. The related columns are excluded from inserts and updates. Disabling this parameter generally improves performance, but there is one case where it could have a negative effect. If multiple rows are being passed in as a single XML, and each row has different columns set (user entered with many optional fields), there is no benefit from batch writing, as each insert or update is different.

- Increase `MaxRaiseSize`

  The `MaxRaiseSize` parameter indicates the maximum number of XML records that can be raised at a time to the BPEL engine. For example, if you set `MaxRaiseSize` = 10, then 10 database records are raised simultaneously. On an inbound read, for example, you can set `MaxRaiseSize` = 0 (unbounded) which means that if you read 1000 rows, you can create one XML with 1000 elements. These elements are passed through a single Oracle BPEL Process Manager instance. A merge on the outbound side can then take all 1000 in one group and write them all at once with batch writing. Use the `MaxRaiseSize` parameter for publishing large payloads.

- Increase `MaxTransactionSize`

  This property controls the number of records processed per transaction by each thread. If set to a large value such as 1000, turning on the `UseBatchDestroy` option could have a negative impact on performance. Setting a large `MaxTransactionSize` and a small `MaxRaiseSize` could also have negative impact on performance. Consider maintaining up to a 10:1 ratio in a synchronous scenario. Ideally, you should consider increasing `MaxRaiseSize` until it is a 1:1 ratio.

- Enable `UseBatchDestroy`

  This property controls how the processed records are updated (ex: `Deleted` for `DeletePollingStrategy`, `MarkedProcessed` for `LogicalDeleteStrategy`). If set, only one update/delete is executed for all the rows that are part of that transaction. The number of rows in a transaction is controlled by the `MaxTransactionSize` option. Note that this may not always offer an improvement because, by default, batch writing is used, which also ends up in a single round trip to the database.

- Enable `Batch Reading`

  Batch reading of one-to-many and one-to-one relationships is on by default. You can also use joined reading for one-to-one relationships instead, which may offer a slight improvement.

- Disable `Delete Polling Strategy`

  Avoid the delete polling strategy because it must individually delete each row. The sequencing polling strategy can destroy 1000 rows with a single update to a helper table. Note that a `LogicalDelete` is also better than Delete, as updates are typically faster than deletes. To maintain performance, however, ensure that you have indexed the table. If you have not indexed, you can keep the total number of rows small by using deletes. In some instances deletes may be faster as the cost of a full table scan is negligible.

- Use `Distributed Polling`

  Distributed polling enables you to configure polling for scalability. For more information, see "Scalability" in *Oracle Fusion Middleware User's Guide for Technology Adapters*.

- Use `Synchronous Processes`

  On BPEL you can configure Database Adapter processes to be synchronous. You can also create sequential routing rules in Mediator. This can improve throughput in database-to-database scenarios, as there is less instance processing impact.

- Use `Insert`

The insert operation is the most performant because it uses no existence check and has no extra performance impact associated with it. There are no reads, only writes. If you know that you are inserting most of the time, use insert, and catch a unique key constraint SQL exception inside your BPEL process, which can then perform a merge or update instead.

To monitor performance, you can enable debug logging and then watch the SQL for various inputs.

- Disable `Merge`

  Merge executes one extra SELECT per related table. The SELECT is used to determine whether each row should be inserted or updated. If the row is updated, the update performed is minimal. If no rows have changed, nothing is updated.

- Use `Connection Pooling`

  The adapter should also point to a tuned data source connection pool. Tuning the connection pool is important because creating and tearing down database connections can impact performance.

- Use `Attribute Filtering`

  On the Attribute Filtering page of the Adapter Configuration Wizard you can choose which fields to map to the XML and vice versa. You can improve performance by deselecting columns that are not needed for your particular business case, especially large columns like LOBs.

- Use `Native Sequencing`

  If you are using the XSL functions to assign primary keys to records, consider using the built-in native sequencing support in the adapter. Sequencing support obtains and caches 50 keys at a time by default. Caching improves performance by reducing the number of round trips. The chunk size can be controlled incrementally by modifying the `sequencePreallocationSize` connector property.

- Do not use primary or foreign keys on the database

  Using primary and foreign keys can impact performance. Avoid using them when possible.

- JDBC Driver Class

  The default JDBC driver class used to create the physical database connections in the connection pool is `oracle.jdbc.xa.client.OracleXADataSource`. Changing the driver to `oracle.jdbc.OracleDriver` may provide some performance improvement.

  For more information on tuning the JDBC drivers, see "Third Party JDBC Driver and Database Connection Configuration" in *Oracle Fusion Middleware User's Guide for Technology Adapters*.

14.3.2 Existence Checking

One method of performance optimization for `merge` is to eliminate check database existence checking. The existence check is marginally better if the row is new, because only the primary key is returned, not the entire row. Due to the nature of merge, however, if the existence check passes, the entire row must be read to calculate what changed. Therefore, for every row to be updated, you see one extra round trip to the database during merge.

Use check cache on the root descriptor/table and any child tables if A is master and B is a privately owned child. If A does not exist, B cannot exist. And if A exists, all of its child tables are loaded as part of reading A.

Note: One way to prevent merge from performing an existence check for every record, when you know that an insert is required, is to set the primary key to null.

14.4 Oracle Socket Adapter Tuning

This section describes performance tuning for Oracle Socket Adapter. Performance can be optimized for the Oracle Socket Adapter using Connection Pool if the socket server you are connecting to does not close the socket with each interaction. Connection pool lets you use a socket connection repeatedly, avoiding the overload of creating a new socket for each interaction.

Note: The Connection Pool feature is applicable to outbound interactions only. For more information on Socket Adapters, see "Oracle JCA Adapter for Sockets" in *Oracle Fusion Middleware User's Guide for Technology Adapters*

In order to enable the connection pool feature for the Oracle Socket Adapter, the `KeepAlive` connection factory property must be set to `True`. This connection property can be modified using the Connection Pool tab of Oracle WebLogic Server Administration Console.

For instructions on modifying the Oracle Socket Adapter connection pooling, see "Configuring Oracle Socket Adapter Connection Pooling" in *Oracle Fusion Middleware User's Guide for Technology Adapters*.

14.5 Oracle SOA JMS Adapter Tuning

This section describes some of the properties that can be set for the Oracle SOA JMS Adapter to optimize performance. See "Introduction to the Oracle JMS Adapter" in the *Oracle Fusion Middleware User's Guide for Technology Adapters* for more information.

14.5.1 adapter.jms.receive.threads Property

To improve performance, the `adapter.jms.receive.threads` property can be tuned for an adapter service. The default value is 1, but multiple inbound threads can be used to improve performance. When specified, the value of `adapter.jms.receive.threads` is used to spawn multiple inbound poller threads.

For example:

```
<service name="dequeue" ui:wsdlLocation="dequeue.wsdl">
<interface.wsdl
interface="http://xmlns.oracle.com/pcbpel/adapter/jms/textmessageusingqueues/textm
essageusingqueues/dequeue%2F#wsdl.interface(Consume_Message_ptt)"/>
<binding.jca config="dequeue_jms.jca">
<property name="adapter.jms.receive.threads" type="xs:string"
many="false">10</property>
</binding.jca">
</service>
```

14.6 Oracle AQ Adapter Tuning

This section describes Oracle AQ Adapter tuning configurations.

14.6.1 adapter.aq.dequeue.threads Property

To improve dequeue performance 'adapter.aq.dequeue.threads' property can be set for an adapter service. Default value is 1 but multiple inbound threads can be used to improve performance. The value of property ‘adapter.aq.dequeue.threads’ is used to spawn multiple inbound poller threads.

For example:

```
<service name="dequeue" ui:wsdlLocation="dequeue.wsdl">
<interface.wsdl
interface="http://xmlns.oracle.com/pcbpel/adapter/aq/raw/raw/dequeue/#wsdl.interfa
ce(Dequeue_ptt)"/>
<binding.jca config="dequeue_aq.jca">
<property name="adapter.aq.dequeue.threads" type="xs:string"
many="false">10</property>
</binding.jca>
</service>
```

14.7 Oracle MQ Adapter Tuning

The Oracle MQ Series Adapter supports the scalability feature for inbound operations only. Oracle MQ Series Adapter provides the parameter to control the number of threads that dequeue the messages from the inbound queue.You must specify the following property in the.jca file:

```
InboundThreadCount='N'
```

In the example above *N* is the number of threads that you want to span to dequeue the messages from the inbound queue.

15

Oracle Business Activity Monitoring Performance Tuning

This chapter describes how to tune the Oracle Business Activity Monitoring (BAM) dashboard application for optimal performance. Oracle BAM provides the tools for monitoring business services and processes in the enterprise.

This chapter discusses useful parameters that can be modified to enhance the overall performance of BAM:

- Section 15.1, "About Oracle Business Activity Monitoring"
- Section 15.2, "Oracle BAM Tuning Considerations"

15.1 About Oracle Business Activity Monitoring

Oracle Business Activity Monitoring (BAM) provides the tools for monitoring business services and processes in the enterprise. It allows correlating of market indicators to the actual business process and to changing business processes quickly or taking corrective actions if the business environment changes. Oracle BAM also provides the necessary tools and run-time services for creating dashboards that display real-time data inflow and define rules to send alerts under specified conditions.

For more information see *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*.

15.2 Oracle BAM Tuning Considerations

The following sections provide Oracle BAM tuning considerations that can be used to address performance issues:

- BAM Server Tuning
- BAM Dashboard Tuning
- BAM Database Tuning
- Internet Browser Tuning
- Enterprise Message Source Tuning

15.2.1 BAM Server Tuning

The following tuning configurations can be used to improve performance of the BAM Server:

15.2.1.1 Set the ViewSetSharing and ElementCountLimit Parameters

The `ViewSetSharing` parameter can be set to `TRUE` or `FALSE` in the BAM server configuration file. This parameter enables view set sharing when possible. Typically a particular view set can be shared with other users if they are trying to access the same dashboard, if the view sets are not dissimilar due to factors like row level security or prompts/parameters tied to filters.

Consider setting the `ViewSetSharing` parameter to `TRUE` so that Active Data Cache (ADC) can reuse the same viewset and snapshot and avoid creating more viewsets. This reduces the BAM server resource usage and improves user response time.

If this parameter is turned on, it does not always guarantee that ADC can reuse the existing viewset. If there have been too many changes to the underlying snapshot for the existing viewset, ADC may choose to create new viewset instead.

The ReportCache parameter used to determine if there have been too many changes is `ElementsCountLimit`. This defines the number of changes to the snapshot used by Report Cache to do the determination. In cases where the active data comes in at a fast rate, try to set this parameter to a large number so that ADC can use view sharing at the expense of more server CPU usage. The default value of `ElementsCountLimit` is 50.

15.2.1.2 Enable the Async Servlet

During periods of higher active data rates, the browser uses more memory. To prevent potential impacts to performance, consider providing more memory on the client machine. To do this, set the `UseAsynchServlet=TRUE` for the BAM dashboard application.

The BAM dashboard application uses the Async servlet feature so that the BAM server does not bind a specific thread to a specific user request. This provides for better server-side system resource usage.

This parameter can be turned off by adding `UseAsynchServlet=FALSE` in the server configuration file. During debugging, consider turning it off to make the process easier.

Otherwise this should always be turned on, which is the default.

See "Creating the Dashboard View" in *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*.

15.2.2 BAM Dashboard Tuning

This section provides information on tuning the BAM dashboard for performance.

15.2.2.1 Tune the Active Data Retrieval Interval

The Active Data Retrieval Interval parameter controls the rate in milliseconds at which the Oracle BAM Active Data Cache (ADC) pushes events to the Oracle BAM Report Server. This is one of the factors that can affect the frequency of viewing active events on the dashboard page. Increasing this interval reduces the load on the Oracle BAM Server. Note that larger intervals increase the likelihood of multiple updates in the dashboard collapsing into a single update.

The default `ADCPushInterval` value is 1 second. You can override the default `ADCPushInterval` value within a particular report using the Active Data Retrieval Interval property in Active Studio.

For more information on using Active Studio, see "Getting Started With Oracle BAM Active Studio" in *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*.

15.2.3 BAM Database Tuning

To achieve the best performance for Oracle Business Activity Monitoring, consider maintaining a database on its own hardware dedicated to the Oracle Business Activity Monitoring system. General database administration practices, as described in the *Oracle Database Performance Tuning Guide*, also apply to a database dedicated to Oracle Business Activity Monitoring.

For more information on general database configurations, see Section 2.6, "Tune Database Parameters".

15.2.4 Internet Browser Tuning

This section provides performance tuning configurations for Internet browsers:

15.2.4.1 Set iActiveDataScriptsCleanupFactor

BAM sends active data in <script> blocks to the browser over a persistent connection. In some cases, the browser does not free up the memory used by the <script> blocks. This can impact dashboard performance over time.

The `iActiveDataScriptsCleanupFactor` parameter provides a solution for these memory leaks. A periodic browser refresh is forced after receiving the specified number of characters. The issue may become apparent when active data is being sent to the dashboard at a fast pace. You may need to increase this value further for particularly high rates of data such as when active data is coming to the dashboard at a rate of 25 events per second or greater. Ultimately the value you set depends on factors like your data, number of views, number of viewsets, ADCPushinterval, and so on). You can monitor the browser's memory consumption to help determine an appropriate value.

If performance continues to be an issue, consider increasing the value for this parameter. For example, set the vale to 2 or 3 times the default value if active data is predicted to increase. The default value for this parameter is 1048576 bytes. The default value often prevents frequent reconnects and prevents CPU/memory on the client machine from creeping up too high.

15.2.4.2 Set Browser Cache Settings

If you are using Microsoft Internet Explorer, consider setting the Browsing History Settings to "Automatic." See the Microsoft Internet Explorer online help for more information.

15.2.5 Enterprise Message Source Tuning

BAM Enterprise Message Source (EMS) provides inbound JMS connectivity to BAM. After setup, a BAM EMS instance can monitor JMS queues/topics and read data from them. Each EMS instance is configured to publish data to a single Data Object in BAM Server. The Enterprise Message Source supports four types of operations: Insert, Update, Upsert, or Delete. Two types of JMS messages are supported: MapMessage and TextMessage.

15.2.5.1 Message Batching

The EMS batching process clubs messages into one single message before it is sent to BAM EMS. This feature enables the sender to send all messages in one batch over JMS. The batching process can improve network performance by limiting the number of round trips from the sender to JMS server to BAM EMS.

16

User Messaging Service Performance Tuning

This chapter describes tips for tuning the User Messaging Service. It contains the following sections:

- Section 16.1, "About Oracle User Messaging Services"
- Section 16.2, "Basic Tuning Considerations"
- Section 16.3, "Database Tuning for Optimal Throughput"

16.1 About Oracle User Messaging Services

Oracle User Messaging Service enables users to receive notifications sent from SOA applications that are developed and deployed to the Oracle WebLogic Server using Oracle JDeveloper.

At the application level, there is notification activity for a specific delivery channel (such as SMS or E-Mail). For example, when you build a SOA application that sends e-mail notification, you drag and drop an Email Activity component from the JDeveloper Component Palette to the appropriate location within a workflow. The application connects then sends notifications.

For more information on Oracle User Messaging Service, see *Oracle WebLogic Communication Services Administrator's Guide*, *Oracle WebLogic Communication Services Developer's Guide*, and the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

16.2 Basic Tuning Considerations

Depending on your User Messaging usage and performance issues, you may consider tuning the following:

- SMPP Driver Performance Tuning
- Email Driver Polling Frequency

16.2.1 SMPP Driver Performance Tuning

Short Messaging Peer-Peer Protocol (SMPP) messaging drivers can be configured using Enterprise Manager. One of the key parameters for optimizing SMPP performance is `WindowSize`. This is especially important when the SMPP driver is connected to a remote SMSC and there is high network latency between the two elements. Configuring the `WindowSize` parameter enables the SMPP driver to send several requests to the Short Messaging Service Center (SMSC) before waiting for an

acknowledgment. Without windowing (i.e., a WindowSize of 1), the driver must wait for a synchronous acknowledgment from the SMSC before sending the next message. With windowing, more messages can be sent per network round-trip, allowing a higher overall throughput.

To take advantage of an increased `WindowSize`, the number of MDB threads for the driver must be correspondingly increased. The two values should be matched so that driver threads can process and send messages before waiting for the requests to be acknowledged. Increasing the two values may improve performance, but only up to the point at which network latency no longer dominates the sending rate. Also, the maximum allowed value for the `WindowSize` is normally defined as a service policy by the SMSC operator.

For more information, see "Configuring Oracle User Messaging Service" in *Oracle WebLogic Communication Services Administrator's Guide*.

16.2.2 Email Driver Polling Frequency

For Email drivers, the "`CheckMailFreq`" configuration parameter defines how frequently the driver checks for incoming emails. For example, a value of "30" means the driver checks the configured inbox every 30 seconds. This parameter can influence performance; checking more frequently enables the driver to keep up with a higher incoming email load, but can impact performance due to frequent IMAP or POP3 operations. Default value is 30 seconds.

16.3 Database Tuning for Optimal Throughput

User Messaging Service stores messaging state such as sent and received messages and delivery status information in the database. Therefore, database and data source tuning may have an effect on messaging throughput. The connection pool size for the data sources can be tuned for higher load levels, but the defaults are sufficient for most cases.

For general database tuning considerations, see Section 2.6, "Tune Database Parameters".

Part V

Identity Management Suite Components

This part describes configuring Oracle Identity Management Suite components to improve performance. The Oracle Identity Management products enable you to configure and manage the identities of users, devices, and services across diverse servers, to delegate administration of these identities, and to provide end users with self-service privileges. These products also enable you to configure single sign-on across applications and to process users' credentials to ensure that only users with valid credentials can log into and access online resources.

It contains the following chapters:

- Chapter 17, "Oracle Internet Directory Performance Tuning"
- Chapter 18, "Oracle Virtual Directory Performance Tuning"
- Chapter 19, "Oracle Identity Federation Performance Tuning"
- Chapter 20, "Oracle Fusion Middleware Security Performance Tuning"

17

Oracle Internet Directory Performance Tuning

This chapter provides guidelines for tuning and sizing an Oracle Internet Directory installation. It contains these topics:

- Section 17.1, "About Oracle Internet Directory"
- Section 17.2, "Introduction to Tuning Oracle Internet Directory"
- Section 17.3, "Basic Tuning Recommendations"
- Section 17.4, "Advanced Configurations"
- Section 17.5, "Low-Priority Tuning Recommendations"
- Section 17.6, "Specific Use Cases"
- Section 17.7, "Optimizing Searches"
- Section 17.8, "Evaluating Performance on UNIX and Windows Systems"
- Section 17.9, "Obtaining Recommendations by Using the Tuning and Sizing Wizard"
- Section 17.10, "Updating Database Statistics by Using oidstats.sql"
- Section 17.11, "Setting Performance-Related Replication Configuration Attributes"
- Section 17.12, "Modifying Performance-Related System Configuration Attributes"
- Section 17.13, "Setting Garbage Collection Configuration Attributes"

17.1 About Oracle Internet Directory

Oracle Internet Directory is Oracle's Lightweight Directory Application Protocol (LDAP) version 3 Directory Server. Oracle Internet Directory is highly scalable, available, and manageable. It has a multi-threaded, multi-process, multi-instance process architecture with Oracle Database as the directory store. This unique physical architecture enables Oracle Internet Directory to be deployed on several hardware architectures including Symmetric Multi-Processor (SMP), Non-Uniform Memory Access (NUMA) and Cluster hardware. Oracle Internet Directory's physical architecture enables linear performance scalability with hardware resources and numerous high availability configurations.

For more information see *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

17.2 Introduction to Tuning Oracle Internet Directory

Note: Oracle Internet Directory's out of box configuration is not optimal for most production or test deployments. You must follow at least the steps listed in Section 17.3, "Basic Tuning Recommendations" to achieve optimal performance and availability.

See Also:

- Section 17.9, "Obtaining Recommendations by Using the Tuning and Sizing Wizard.".
- The "Troubleshooting Directory Performance" appendix in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

Many of the recommendations in this chapter require changes to Oracle Internet Directory system configuration attributes and replication configuration attributes.

See Also:

- The "Managing System Configuration Attributes" chapter of *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- The "Managing Replication Configuration Attributes" chapter of *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- The "Attribute Reference" chapter of *Oracle Fusion Middleware User Reference for Oracle Identity Management*

for more information about Oracle Internet Directory configuration attributes.

17.3 Basic Tuning Recommendations

Tuning is the adjustment of parameters to improve directory performance. The default Oracle Internet Directory configuration must be tuned in almost all deployments. Please review the requirements and recommendations in this section carefully.

17.3.1 Database Parameters

Some good minimum values for Oracle Database instance parameters are given here:

Table 17–1 Minimum Values for Oracle Database Instance Parameters

Parameter	Value	Notes
`sga_target` and `sga_max_size`	1700M for 32-bit systems	Applicable when SGA Auto Tuning using `sga_target` and `sga_max_size` is being used. Especially important for `bulkdelete` performance. A higher value may be required if the directory size exceeds 1 million entries or a high rate of I/O is observed. In case of 64-bit systems, one can go up to 60-70% of the RAM available for the Oracle Database on the box.

Table 17–1 (Cont.) Minimum Values for Oracle Database Instance Parameters

Parameter	Value	Notes
db_cache_size	1200M for 32-bit systems.	Applicable when SGA Auto Tuning using sga_target and sga_max_size is not being used. (SGA auto tuning using sga_target and sga_max_size is recommended instead of this parameter.) A higher value may be required if the directory size exceeds 1 million entries or a high rate of I/O is observed. In case of 64-bit systems, one can go up to 60-70% of the RAM available for the Oracle Database on the box.
shared_pool_size	300M	Applicable when SGA Auto Tuning using sga_target and sga_maxsize is not being used
session_cached_cursors	100	
processes	500	
pga_aggregate_target	300M	Before performing a large bulkload operation, set this to 1-4GB, if sufficient RAM is available. Set it back after the operation has completed
job_queue_processes	1 or more.	Tune this parameter only if you are using Oracle Database Advanced Replication-based multimaster replication
max_commit_propagation_delay	99 or lower	Tune this parameter only in RAC Database deployments, RDBMS v10.1.

See the *Oracle Database Performance Tuning Guide* for information on setting Oracle Database instance parameters.

17.3.2 LDAP Server Attributes

The recommendations in this section are summarized in Table 17–2.

- Tune the number of processes and threads for the Oracle Internet Directory server instance that services LDAP application traffic. This has a major impact on overall performance. See the recommended settings for orclmaxcc and orclserverprocs in Table 17–2.
- Disable change log generation if you are not deploying either replication or Oracle Directory Integration Platform. Set the attribute orclgeneratechangelog to 0.
- Skip referrals in LDAP searches if you have no referral entries in the directory. Set orclskiprefinsql to 1. This has a major impact on performance.
- Close idle LDAP connections after a period of time instead of leaving them open. This prevents the unnecessary buildup of connections. For example, you can set orclldapconntimeout to 60 minutes.

  As of 10*g* (10.1.4.0.1), you can only set this for users who are not configured for operation statistics tracking. Connections by users configured for statistics collection do not time out as per this setting.

> **See Also:** "Configuring a User for Statistics Collection by Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

- If no clients require detailed MatchDN information when the Base DN of an LDAP search operation is not present in the directory, disable it. Change `orclmatchdnenabled` to 0.

The following values are appropriate for most deployments:

Table 17–2 LDAP Server Attributes to Tune

Attribute	Default	Recommended Value	Notes
`orclmaxcc`	2	10	Server restart required.
`orclserverprocs`	1	Number of CPU sockets on Oracle Internet Directory node	
`orclskiprefinsql`	0	1	This change is highly recommended. Do not change if you have LDAP referral entries. LDAP referral entries are not common. Server restart required.
`orclgeneratechangelog`	1	0	Disable change log generation only if you do not deploy either replication or Oracle Directory Integration Platform.
`orclldapconntimeout`	0 (no timeout)	Varies, 60 is reasonable	Users configured for statistics tracking do not time out.
`orclmatchdnenabled`	1	0	Disable only if no application needs detailed MatchDN information when base DN of a search is not present.

For information about configuring `orclserverprocs`, `orclldapconntimeout`, and `orclmatchdnenabled` with Oracle Enterprise Manager Fusion Middleware Control, see Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control."

For information about configuring `orclskiprefinsql` or `orclmatchdnenabled` with Oracle Enterprise Manager Fusion Middleware Control, see Section 17.12.2, "Modifying Shared Attributes by Using Fusion Middleware Control."

For information about configuring these attributes, as well as `orclgeneratechangelog`, from the command line, see Section 17.12.3, "Modifying Attributes by Using ldapmodify."

17.3.3 Database Statistics

If you use LDAP commands to add a large number entries to Oracle Internet Directory, it can affect directory performance. If this occurs, update the database statistics. See Section 17.10, "Updating Database Statistics by Using oidstats.sql."

Typically, you only need to do this when you add entries in bulk for the first time after Oracle Internet Directory installation. You do not need to do it again because the database statistics are updated nightly automatically. If, however, you suddenly experience slow LDAP operations, without a corresponding change in data footprint, consider running `oidstats.sql` once to see if that improves performance. The

impact may be due to changes in database SQL execution plans, which oidstats.sql can help to improve.

> **See Also:** *Oracle Database Performance Tuning Guide* for information about SQL tuning.

You do not need to update database statistics if you use the `bulkload` tool to add the entries. The `bulkload` command automatically updates the database statistics.

17.4 Advanced Configurations

After you have performed the modifications recommended in the previous section, you can make additional changes that are specific to your deployment. Consider carefully whether the recommendations in this section are appropriate for your environment.

17.4.1 Replication or Oracle Directory Integration Platform

When you deploy Oracle Internet Directory with the Oracle Directory Integration Platform or with replication, you can improve performance by having a dedicated LDAP server instance for those two servers. This allows the default Oracle Internet Directory LDAP instance to serve the LDAP application traffic and the second instance to serve LDAP requests from the replication and Oracle Directory Integration Platform servers.

1. Create an additional server instance, as described in the chapter "Managing Oracle Internet Directory Instances" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.
2. Set `orclmaxcc` to 10 and `orclserverprocs` to 1 in the new instance configuration.
3. Restart the server, as described in the chapter "Managing Oracle Internet Directory Instances" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.
4. Set the SSL and non-SSL ports used by the new instance and configure the replication and Oracle Directory Integration Platform to point to them.

To configure `orclmaxcc` and `orclserverprocs`, see Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control" and Section 17.12.3, "Modifying Attributes by Using ldapmodify."

> **Note:** In an Oracle Internet Directory Cluster configuration (rack-mounted or multi-box), the replication server must be started on one hardware node only. The LDAP server instance dedicated to replication must be started on the same node. The Oracle Directory Integration Platform server can be on a different node.

17.4.2 Replication Server Configuration

The following recommendations can be useful when replication traffic is heavy. Be sure you understand the trade-offs before making these changes. The recommended values are summarized in Table 17–3.

- If you are deploying a single master with read-only replica consumers, you may reduce performance impacts by turning off conflict resolution. To do so, change the value of `orclconflresolution` to 0.
- If the supplier is a bottleneck, increase `orclthreadspersupplier` on the supplier. You can also increase `orclthreadspersupplier` at the consumer if is a bottleneck, but be aware that increased parallelism causes race conditions in the application of changelogs, resulting in more human intervention queue (HIQ) changes.
- Decrease `orclchangeretrycount` so that new changelogs get more resources. If there are conflicts, however, this increases the human intervention queue (HIQ) changes.
- Change `orclupdateschedule` to 0 to make the server process changelogs immediately, instead of at the default, 60-second intervals. Do this on both the supplier and consumer.
- Increase the `orclhiqschedule` to a higher value. For example, if accessing the human intervention queue (HIQ) four times a day is sufficient and appropriate for your deployment, set the `orclhiqschedule` to 21600 seconds (6 hours).

Table 17–3 summarizes these recommendations.

Table 17–3 Replication Attributes

Attribute	Default	Recommended Value	Notes
`orclthreadspersupplier`	transport=1 apply=5	Set transport threads to 1 and apply threads to 10 or greater	Most useful if the supplier is the bottleneck.
`orclchangeretrycount`	10	4	Provides more resources to changelogs but might increase HIQ.
`orclupdateschedule`	60 seconds	0	Causes changelogs to be processed immediately
`orclhiqschedule`	600 seconds	21600 seconds	Provides more resources to process new changes.
`orclconflresolution`	1	0	Change only if you are deploying a single master with read-only replica consumers.

See Section 17.11, "Setting Performance-Related Replication Configuration Attributes" for information on setting these replication attributes.

17.4.3 Garbage Collection Configuration

By default, Oracle Internet Directory runs database jobs to purge change logs, server manageability statistics, and other data beginning at midnight, with each job starting 15 minutes after the previous one. You can change this configuration to suite your deployment needs by modifying the parameters shown in Table 17–4.

Table 17–4 Garbage Collection Configuration Parameters

Parameter	Value	Notes and References
`orclpurgetargetage`	Less than 10days (240 hours)	Only if there is no requirement to retain change logs
`orclpurgeinterval`	6–12 hours	

You can modify these attributes by using `ldapmodify` or Oracle Directory Services Manager. See Section 17.13, "Setting Garbage Collection Configuration Attributes."

17.4.4 Oracle Internet Directory with RAC Database

As described in Section 17.4.2, "Replication Server Configuration", you can have a dedicated LDAP server for Oracle Directory Integration Platform and replication, in addition to the default server. In an Oracle Internet Directory Cluster, start the default LDAP instance on all Oracle Internet Directory nodes, but start the dedicated instance only on the node where Oracle Directory Integration Platform and replication are running.

Consider carefully which database instance Oracle Internet Directory should connect to:

- You can configure the Oracle Internet Directory for load balancing between Oracle Database instances in the cluster, or failover mode.
- If you use a dedicated LDAP server instance for replication and Oracle Directory Integration Platform, you can configure the connection strings of that instance for failover. You would use the following in `tnsnames.ora`:

  ```
  (FAILOVER=ON)(LOAD_BALANCE=OFF)
  ```

- When performing a bulk operation, such as `bulkload`, connect the tool to just one Oracle Database instance for the entire operation.
- Configure Oracle Internet Directory instances as follows:
  - One Oracle Internet Directory instance on each of the nodes to service LDAP application traffic
  - An instance of the Oracle Internet Directory replication server and Oracle Directory Integration Platform server on one node

17.4.5 Password Policies and Verifier Profiles

Oracle Internet Directory has password policies and password verifier profiles enabled out of box. If Oracle Internet Directory is not required to enforce password policies in a given deployment, then the password policies can be disabled. The password verifier profiles enabled out of box control the generation of certain password verifiers required by Oracle products like Enterprise User Security and Oracle Collaboration Suite. If Oracle Internet Directory is not being deployed for other Oracle products, you can disable all the password verifier profiles.

You can disable password policies and password verifiers by using Oracle Directory Services Manager or `ldapmodify`.

See Also:

- The "Managing Password Policies" chapter in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.
- The "Managing Password Verifiers" chapter in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

17.4.6 Server Entry Cache

The Oracle Internet Directory server entry cache enables LDAP entries to be cached on the Oracle Internet Directory server process heap for better performance. Configuring the entry cache provides benefits if, and only if, all or most entries can be cached.

Note: The server entry cache is beneficial for small directory deployments only! Some of the tuning recommendations here contradict the tuning recommendations in the earlier sections. Review the applicability of entry cache to a given deployment and incorporate the tuning mentioned in this section only if all considerations enumerated here are met.

17.4.6.1 When to Use the Entry Cache

Consider using Oracle Internet Directory Server Entry Cache only under the following conditions:

- The total number of entries in Oracle Internet Directory can be fully or mostly cached. This is usually the case for deployments with fewer than 500K entries in Oracle Internet Directory on a 32-bit system
- The number of concurrent clients is low, typically less than 100
- You are not using a cluster configuration
- You do not require the LDAP server instance to be multiprocess.
- You expect a very low update rate, especially on group entries.
- You are not using a second, dedicated LDAP server instance for replication or Oracle Directory Integration Platform
- Very few applications are using Oracle Internet Directory
- You have no large binary values or large group entries, and updates on binary and group entries are infrequent.

17.4.6.2 Benefits of Using the Entry Cache

Benefits of using the entry cache include:

- LDAP search operations with subtree and one-level scope are about twice as fast.
- LDAP search operations with base scope are about five times as fast.

These benefits apply only when all or most entries can be cached. A cache miss is more expensive than disabling the entry cache.

17.4.6.3 Values for Configuring the Entry Cache

You can configure and optimize the server entry cache by setting the values shown in Table 17–5.

Table 17–5 Server Entry Cache Configuration

Attribute	Default	Recommended Value	Notes
orclmaxcc	2	Total number of processor cores on the node	Restart the server after changing this attribute.
orclserverprocs	1	1	For values greater than 1, entry cache is automatically disabled. Restart the server after changing this attribute.
orclecacheenabled	1	1	
orclecachemaxsize	200000000 Bytes	Total size of the directory, in bytes	Estimate three times the size of the entries in LDIF format
orclecachemaxentries	100000	Total number of entries in the DIT	
orclecachemaxentsize	1000000	Size, in bytes, of the largest entry in the DIT	The largest entry is usually a group entry or an entry with binary attribute values.

For example, if the total size of the DIT is 300K and total size of 300K entries in LDIF format is 500M, you would set orclecacheenabled to 1, orclecachemaxsize to 1500000000, and orclecachemaxentries to 300000. If the size of the largest group entry or entry with binary value is 10M, you would set orclecachemaxentsize to 10000000.

To configure the attributes, see Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control" and Section 17.12.3, "Modifying Attributes by Using ldapmodify."

17.4.7 Tuning Security Event Tracking

The instance-specific configuration entry attributes orcloptrackmaxtotalsize and orcloptracknumelemcontainers control how much memory is used for security event tracking.

The attribute orcloptrackmaxtotalsize specifies the maximum number of bytes of RAM that security events tracking can use for each type of operation. If the Directory Server exceeds this limit for information collected for an operation, the server stops collecting new information and records appropriate messages in server log files. For the compare operation, the Directory Server uses twice the value of the attribute, which is the combined amount of information about users performing compare operation and users whose passwords are being compared. The default value of orcloptrackmaxtotalsize is 100000000 Bytes, which should be sufficient for most deployments. It can be increased to 200MB. For information about modifying orcloptrackmaxtotalsize, see the instance-specific configuration attribute examples in Section 17.12.3, "Modifying Attributes by Using ldapmodify."

The attribute orcloptracknumelemcontainers allows you to choose the number of in-memory cache containers to be allocated for security event tracking in the Oracle Internet Directory server. There are two subtypes for this attribute. They are 1stlevel and 2ndlevel. The 1stlevel subtype is for setting the number of in-memory cache containers for storing information about users performing

operations. The `2ndlevel` subtype, which is applicable only to compare operation, sets the number of in-memory cache containers for information about the users whose userpassword is compared and tracked when detailed compare operation statistics is programmed.

The default value of both subtypes is `256`. The appropriate values for these subtypes depend on the number of users in your environment and the number of applications used to access the directory, as follows:

- In a deployment where several applications perform operations on behalf of a large number of end users, set 1stlevel proportional to the number of applications, plus a few hundred more for end users directly accessing the directory. Then set 2ndlevel proportional to the number of end users.
- In a deployment where end users themselves perform the operations, set 1stlevel proportional to the number of end users, then set 2ndlevel to a small value, such as 25.
- A typical proportional value is one fifth. Proportions between one tenth and one half are reasonable in most environments.

If your deployment requires it, set the values for orcloptracknumelemcontainers only when security events collection is turned on.

17.5 Low-Priority Tuning Recommendations

This section describes attributes that can sometimes improve performance, but are considered low-priority.

17.5.1 Number of Entries to be Returned by a Search

The attribute `orclsizelimit` controls the maximum number of entries to be returned by a search. The default value is 10000. Setting it very high impacts server performance. It also plays a role in limiting the maximum number of changelogs the replication server can process at a time.

See Section 17.12.3, "Modifying Attributes by Using ldapmodify."

17.5.2 Enabling the Group Cache

The instance-specific subentry attribute `orclenablegroupcache` controls whether privilege groups and ACL groups are cached. Using this cache can improve the performance of access control evaluation for users.

Use the group cache when a privilege group membership does not change frequently. If a privilege group membership does change frequently, then it is best to turn off the group cache. It is important to note that computing a group cache may affect performance. The default is 1 (enabled). Change to 0 (zero) to disable.

See Section 17.12.3, "Modifying Attributes by Using ldapmodify."

17.5.3 Timeout for Write Operations

When an LDAP client initiates an operation, then does not respond to the server for a configured number of seconds, the server closes the connection. The number of seconds is controlled by the `orclnwrwtimeout` attribute of the instance-specific configuration entry. The default is 30 seconds.

You can modify `orclnwrwtimeout` by using Fusion Middleware Control or the command line. See Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control."

17.6 Specific Use Cases

This section describes some specific use cases that require additional tuning, in addition to Section 17.3, "Basic Tuning Recommendations"

17.6.1 Bulk Load Operation

If you are planning a large `bulkload` operation, make the following changes:

- Set the database initialization parameter `pga_aggregate_target` to 1-4GB for the duration of the operation, if sufficient RAM is available.
- Increase the database temporary tablespace before loading a large number entries. You need about 1G of temporary tablespace per million entries being loaded. You can free up the tablespace after the operation.

17.6.2 Bulk Delete Operation

If you are planning a large `bulkdelete` operation, perform the following tasks:

- Ensure that the database initialization parameter `sga_target` are tuned as described in Section 17.3.1, "Database Parameters."
- Set the database initialization parameter `log_buffer` to 10M. This can provide additional performance benefit.
- Ensure that you have at least three database redo log files with at least 100MB.
- Ensure that the undo tablespace is at least 1 GB in total size.
- Follow the recommendations about redo logs and undo tablespace in the next section, Section 17.6.3, "High LDAP Write Operations Load."

17.6.3 High LDAP Write Operations Load

If you have a high LDAP write operations load, or if you perform many `bulkdelete` operations, consider tuning the following values:

- Increase the size or number of the database redo log files so that the total size is 1000-1500 MB. Other considerations affect the total size of redo logs.
- Depending on how the disks are configured, it might be beneficial to isolate the redo log files to a dedicated set of disks.
- Increase the undo tablespace size by adding data files to this tablespace. For most deployments, 2-4 GB should suffice.
- Do not use the Oracle Internet Directory server entry cache. See Section 17.4.6, "Server Entry Cache."
- If neither Oracle Internet Directory replication nor DIP is deployed, disable change log generation. See Section 17.4.1, "Replication or Oracle Directory Integration Platform."

Table 17–6 summarizes the redo log and undo tablespace recommendations provided in this section.

Table 17–6 Redo Log and Undo Tablespace Values

Attribute	Value	Notes
Redo Log	3 logs, 100MB each	Many `bulkdelete` operations.
Redo Log	Total size 1000-15000MB	Large number of write operations.
Undo Tablespace	At least 1GB total	Many `bulkdelete` operations.
Undo Tablespace	2-4 GB	Large number of write operations.

17.7 Optimizing Searches

This section contains these topics:

- Section 17.7.1, "Optimizing Searches for Large Group Entries"
- Section 17.7.2, "Optimizing Searches for Skewed Attributes"
- Section 17.7.3, "Optimizing Performance of Complex Search Filters"

17.7.1 Optimizing Searches for Large Group Entries

Searches for group entries with several thousand attribute values for either the `member` or `uniquemember` attribute can have high latency. If you find the latency unacceptably high, there are steps you can take to reduce it.

The simplest step is to reduce the number of attributes you are searching for. If you do not need to retrieve all the attributes of the group entry, specify required attributes in the search request to optimize the latency.

17.7.1.1 Entry Cache Enabled Configuration

If you still see unacceptable latency, even with required attributes specified, then you can try to cache the large group entry in the entry cache. To do this, increase the value of the `orclEcacheMaxEntSize` attribute in the instance-specific configuration entry:

`cn=`*`componentname`*`,cn=osdldapd,cn=subconfigsubentry`

This attribute controls the maximum size of a cache entry. The default value is 1M. If the size of the large group entry is greater than the value of `orclEcacheMaxEntSize`, change it to a large enough value to ensure that the large group entry is cached.

> **Note:** If you expect frequent updates to large groups, then do not use this tuning methodology. Use the Entry Cache Disabled Configuration.

17.7.1.2 Entry Cache Disabled Configuration.

No action is required. This configuration is enabled by default.

17.7.2 Optimizing Searches for Skewed Attributes

To service a typical search request, the Directory Server sends a SQL statement to the Oracle Database. If a given attribute has very different response times depending on its value, then the attribute is said to be skewed. For example, if searches for `my_attribute=value1` and `my_attribute=value2` have very different response times, then `my_attribute` is said to be a skewed.

You can uniform the response times for searches for such an attribute by adding it as a value of the `orclskewedattribute` attribute, which is in the DSA configuration entry. The DN of the DSA configuration entry is

```
cn=dsaconfig,cn=configsets,cn=oracle internet directory
```

By default, the `objectclass` attribute is listed as a value in the `orclskewedattribute` attribute.

You can change the value of `orclskewedattribute` by using or `ldapmodify`. See Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control" and Section 17.12.3, "Modifying Attributes by Using ldapmodify".

17.7.3 Optimizing Performance of Complex Search Filters

When Oracle Internet Directory receives an LDAP search filter from a client application, it sends the filter to the Oracle Database as an SQL query. Sometimes client applications send filters that include terms that match a large number of entries in the directory. For example, consider the following filter:

```
(&(uid=msmith)(objectclass=inetorgperson)(orclisenabled=TRUE))
```

The terms `(objectclass=inetorgperson)` and `(orclisenabled=TRUE)` in that filter match nearly all entries. It would be very resource-intensive to execute that entire filter in the Oracle Database. To improve performance, you can specify that Oracle Internet Directory execute a portion of that filter in its own memory, rather than in the database. To do that, you use `orclinmemfiltprocess`, an attribute in the DSA configuration entry:

```
cn=dsaconfig,cn=configsets,cn=oracle internet directory
```

When `orclinmemfiltprocess` is configured, the following events occur each time Oracle Internet Directory receives an LDAP search:

1. Oracle Internet Directory removes all the terms that are configured in the `orclinmemfiltprocess` before forming the SQL query.
2. Oracle Internet Directory sends the SQL query to Oracle Database.
3. Oracle Database sends the entries resulting from the SQL query to Oracle Internet Directory.
4. Oracle Internet Directory applies the original filter sent by the client (the terms in `orclinmemfiltprocess`) to those entries in memory.
5. Oracle Internet Directory sends the entries that match that filter to the client.

For example, suppose `orclinmemfiltprocess` is set to `(objectclass=inetorgperson)(orclisenabled=TRUE)`. When Oracle Internet Directory receives the search `(&(uid=msmith)(objectclass=inetorgperson)(orclisenabled=TRUE))`, it sends a filter containing only the parameter `(uid=msmith)` to the database. After Oracle Internet Directory receives entries back from the database, Oracle Internet

Directory itself applies the filter `(objectclass=inetorgperson)(orclisenabled=TRUE)` to those entries.

By default, `orclinmemfiltprocess` is set to the following values:

```
(objectclass=inetorgperson)
(objectclass=oblixorgperson)
(|(!(obuseraccountcontrol=*))(obuseraccountcontrol=activated))
(|(obuseraccountcontrol=activated)(!(obuseraccountcontrol=*)))
(objectclass=*)
(objectclass=oblixworkflowstepinstance)
(objectclass=oblixworkflowinstance)
(objectclass=orcljaznpermission)
(obapp=groupservcenter)(!(obdynamicparticipantsset=*))
(objectclass=orclfeduserinfo)
```

You can change the value of `orclinmemfiltprocess` by using or `ldapmodify`. See Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control" and Section 17.12.3, "Modifying Attributes by Using ldapmodify".

Under some conditions, Oracle Internet Directory ignores `orclinmemfiltprocess` and sends the entire filter to the database. It does this if the filter it receives meets the following conditions:

- It contains only one parameter, that is, one attribute-value pair.
- It contains no filter condition other than those in `orclinmemfiltprocess`
- It contains an OR condition applied to the terms that are in `orclinmemfiltprocess`
- It contains the same terms as in `orclinmemfiltprocess`, but in a different order

The following cases illustrate those conditions. In all of the following cases, `orclinmemfiltprocess` is set to `(objectclass=inetorgperson)(employeetype=Contract)`.

Examples

Case A

```
(&(manager=cn=john doe)(objectclass=inetorgperson)
(employeetype=Contract))
```

Oracle Internet Directory sends the filter `(&(manager=cn=john doe))` to the database.

Case B

```
(&(uid=rmsmith)((objectclass=inetorgperson)(employeetype=Contrac
t)))
```

Oracle Internet Directory sends only `(&(uid=rmsmith))` to the database, then applies the filter `(&(objectclass=inetorgperson)(employeetype=Contract))` to the entries that are returned from the database.

Case C

```
(|(uid=rmsmith)(objectclass=inetorgperson)
(employeetype=Contract))
```

In this filter, the terms that match `orclinmemfiltprocess` are part of an OR condition. Oracle Internet Directory sends the filter, as is, to the database.

Case D

```
(&(uid=rmsmith)(employeetype=Contract)
(objectclass=inetorgperson))
```

Even though some of the terms in this filter match `orclinmemfiltprocess`, they are in a different order, so Oracle Internet Directory sends the whole filter to the database. You could add `(employeetype=Contract)(objectclass=inetorgperson)` to `orclinmemfiltprocess` if you do not want Oracle Internet Directory to send this filter to the database.

Case E

```
(|(&(uid=rmsmith)(sn=smith)(objectclass=inetorgperson)(employeet
ype=Contract))
```

In this filter, the terms that match `orclinmemfiltprocess` are part of an OR condition. Oracle Internet Directory sends the filter, as is, to the database.

Case F

```
(&(|(uid=rmsmith)(sn=smith))(objectclass=inetorgperson)(employee
type=Contract)))
```

Even though this filter contains an OR operator, it is not applied to the terms that match `orclinmemfiltprocess`. Oracle Internet Directory sends `(&(|(uid=rmsmith)(sn=smith)))` to the directory and applies the filter `(&(manager=cn=john doe)(&(objectclass=inetorgperson)(employeetype=Contract))` to the entries that are returned from the database.

Configuring Multiple Filters

If the application is sending multiple filters, and the terms in one filter are a superset of the terms in the other, you must configure `orclinmemfiltprocess` for both values.

For example, suppose the application is sending the following two filters:

```
(&(uid=rmsmith)(objectclass=inetorgperson)(employeetype=Contract
))
```

```
(&(uid=rmsmith)(objectclass=inetorgperson)(employeetype=Contract
)(departmentNumber=627))
```

where `(departmentNumber=627)` matches a lot of entries. You must configure `orclinmemfiltprocess` as follows:

```
(objectclass=inetorgperson)(employeetype=Contract)
```

```
(departmentNumber=627)
```

17.8 Evaluating Performance on UNIX and Windows Systems

Knowledge of the following tools is recommended for Linux, Solaris, and other UNIX-like operating systems:

Tool	Description
top	Displays the top CPU consumers on a system
vmstat	Shows running statistics on various parts of the system including the Virtual Memory Manager
mpstat	Shows an output similar to vmstat but split across various CPUs in the system. This is available on Solaris only.
iostat	Shows the disk I/O statistics from various disk controllers
sar	Collect, report, or save system activity information.

Knowledge of the following tools is recommended for Microsoft Windows:

Tool	Description
Windows Performance Monitor	Provides a customized view of the events in the system
Windows Task Manager	Provides a high level output (like `top` on UNIX) of the major things happening in the system.

Knowledge of the following tools is recommended for the Oracle Database:

- `utlbstat.sql` and `utlestat.sql`, or `statspack`
- The ANALYZE function in the DBMS_STATS package

See Also:

- *Oracle Database Reference* in the Oracle Database Documentation Library for information about `utlbstat.sql` and `utlestat.sql`
- *Oracle Database Performance Tuning Guide* for information about stats package
- *Oracle Database Concepts* in the Oracle Database Documentation Library for information about the ANALYZE function in the DBMS_STATS package

In addition to the operating system tools, the LDAP applications being used in a customer environment must be able to provide latency and throughput measurement.

In addition, the Database Statistics Collection Tool (oidstats.sql), located at `$ORACLE_HOME/ldap/admin`, is provided to analyze the various database 'ods' schema objects to estimate the statistics. See Section 17.10, "Updating Database Statistics by Using oidstats.sql".

17.9 Obtaining Recommendations by Using the Tuning and Sizing Wizard

Oracle Enterprise Manager Fusion Middleware Control provides a convenient tool for tuning and sizing Oracle Internet Directory.

Use the wizard to obtain tuning and sizing recommendations for your system. You can select Tuning, Sizing, or Both. If you select Sizing or Both, you can select Basic or Advanced

Tuning

1. From the Oracle Internet Directory menu, select **Administration**, then **Tuning and Sizing**.
2. Click the **Create** icon to invoke the wizard.
3. On the Type Selection page, change the report name, then select **Tuning**.
4. The wizard presents the following pages: Hardware, Features, Load, Data Characteristics, and Garbage Collection.

   On each page, specify values for the text fields (or use defaults) and Select **Yes** or **No** for each question. Some choices might be greyed out, depending upon your previous choices. Most fields have tool tips that appear when you move the cursor over the field.

   Click **Next** to go to the next page or **Back** to return to the previous page. Click **Cancel** to close the wizard.
5. On the Review page, review the data you entered. Click **Back** to change your specifications or click **Finish** to view the report.
6. The report appears on the bottom right section of the page.

   To download the report, click **Download Report**. To delete the report, click **Delete**.

Sizing

1. From the Oracle Internet Directory menu, change the report name, then select **Administration**, then **Tuning and Sizing**.
2. Click the **Create** icon to invoke the wizard.
3. On the Type Selection page, select **Sizing**.
4. Select **Basic** or **Advanced**.
5. On the Sizing page, specify values for the text fields (or use defaults) and Select Yes or No for each question. Some choices might be greyed out, depending upon your previous choices.
6. Click **Next**.
7. On the Review page, review the data you entered. Click **Back** to change your specifications or click **Finish** to view the report.
8. The report appears on the bottom right section of the page.

   To download the report, click **Download Report**. To delete the report, click **Delete**.

Both

1. From the Oracle Internet Directory menu, change the report name, then select **Administration**, then **Tuning and Sizing**.
2. Click the **Create** icon to invoke the wizard.
3. On the Type Selection page, select **Both**.
4. Select **Basic** or **Advanced**.
5. Click **Next**.

6. The wizard presents the following pages: Sizing, Hardware, Features, Load, Data Characteristics, and Garbage Collection.

   On each page, specify values for the text fields (or use defaults) and Select **Yes** or **No** for each question. Some choices might be greyed out, depending upon your previous choices.

   Click **Next** to go to the next page or **Back** to return to the previous page. Click **Cancel** to close the wizard.

7. On the Review page, review the data you entered. Click **Back** to change your specifications or click **Finish** to view the report.

8. The report appears on the bottom right section of the page.

   To download the report, click **Download Report**. To delete the report, click **Delete**.

17.10 Updating Database Statistics by Using oidstats.sql

To update database statistics, execute the Oracle Internet Directory Database Statistics Collection tool, as follows:

```
ORACLE_HOME/ldap/admin/oidstats.sql
```

You do not need to run `oidstats.sql` if you use the `bulkload` tool to add entries to the database. The `bulkload` command automatically updates the database statistics.

If you load data into the directory by any means other than the bulk load tool (`bulkload`), then you must run `oidstats.sql` after loading. Statistics collection is essential for the Oracle Optimizer to choose an optimal plan in executing the queries corresponding to the LDAP operations. You can run Oracle Internet Directory Database Statistics Collection tool at any time, without shutting down any of the Oracle Internet Directory daemons.

Note: If you do not use the bulkload utility to populate the directory, then you must run the `oidstats.sql` tool to avoid performance impacts.

See Also: The `oidstats.sql` command-line tool reference in *Oracle Fusion Middleware User Reference for Oracle Identity Management*

17.11 Setting Performance-Related Replication Configuration Attributes

To set the replication attributes, you can use either the Replication Wizard in Oracle Enterprise Manager Fusion Middleware Controlor the command line.

The attributes `orclthreadspersupplier`, `orclchangeretrycount`, and `orclconflresolution` are replication configuration set attributes.

See Also:

- "Configure Replication Attributes by Using Fusion Middleware Control" in*Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- "Configuring Attributes of the Replication Configuration Set by Using ldapmodify" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

for information about

The attributes `orclhiqschedule` and `orclupdateschedule` are replication agreement entry attributes.

See Also:

- "Viewing or Modifying an LDAP-Based Replication Setup by Using the Fusion Middleware Control Replication Wizard" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- "Configuring Replication Agreement Attributes by Using ldapmodify" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

See Also:

- "Setting Up a One-Way, Two-Way, or Multimaster LDAP-Based Replication Agreement by Using the Replication Wizard in Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* or information on setting replication attributes by using the Replication Wizard.
- "Configuring Attributes of the Replication Configuration Set by Using ldapmodify" in Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory.

17.12 Modifying Performance-Related System Configuration Attributes

You can set most performance-related system configuration attributes from Oracle Enterprise Manager Fusion Middleware Control or from the command line. This section describes how to do that.

You can also use the Data Browser in Oracle Directory Services Manager to modify system configuration attributes.

See Also: "Managing System Configuration Attributes by Using Oracle Directory Services Manager Data Browser" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

This section contains the following topics:

- Section 17.12.1, "Modifying Instance-Specific Attributes by Using Fusion Middleware Control"
- Section 17.12.2, "Modifying Shared Attributes by Using Fusion Middleware Control"

- Section 17.12.3, "Modifying Attributes by Using ldapmodify"

17.12.1 Modifying Instance-Specific Attributes by Using Fusion Middleware Control

You can configure performance attributes in the instance-specific configuration entry by using the Server Properties page of Oracle Enterprise Manager Fusion Middleware Control. Select **Administration**, then **Server Properties** from the **Oracle Internet Directory** menu, then select the **Performance** tab.

Table 17–7 shows the relationship between fields on the page and configuration attributes.

Table 17–7 Configuration Attributes on Server Properties Page, Performance Tab

Field or Heading	Configuration Attribute
Number of Oracle Internet Directory LDAP Server Processes	`orclserverprocs`
Number of DB Connections per Server Process	`orclmaxcc`
Enable Entry Cache	`orclecacheenabled`
Maximum Entries in Entry Cache	`orclecachemaxentries`
Maximum Entry Size in Cache (byte)	`orclecachemaxentsize`
Maximum Entry Cache Size (MB)	`orclecachemaxsize`
Number of users in privilege group membership cache	`orclmaxconnincache`
LDAP Idle Connection Timeout (sec)	`orcllldapconntimeout`
Oracle Internet Directory server Network Read/Write Retry Timeout (sec)	`orclnwrwtimeout`
Maximum Time in seconds for Server process to respond back to Dispatcher process	`orclMaxServerRespTime`
Number of Dispatcher Threads per Server Process	`orcldispthreads`
Maximum Number of LDAP connections per Server Process	`orclmaxldapconns`
Number of Plugin Threads per Server Process	`orclpluginworkers`
Enable Change Log Generation	`orclgeneratechangelog`

Restart the server after changing `orclserverprocs`, `orclmaxcc`, `orcldispthreads`, or `orclpluginworkers`.

17.12.2 Modifying Shared Attributes by Using Fusion Middleware Control

You configure the performance-related shared attributes in the DSA configuration entry by using the **General** tab of the Oracle Internet Directory **Shared Properties** page of Oracle Enterprise Manager Fusion Middleware Control. Select **Administration**, then **Shared Properties** from the **Oracle Internet Directory** menu.

Table 17–8 shows the relationship between fields on the page and the performance-related configuration attributes.

See Also: "Configuring Shared Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* for information about other fields on the Shared Properties page.

Table 17–8 Performance-Related Attributes on Shared Properties Page, General Tab

Field or Heading	Configuration Attribute
Skip referral for search	`orclskiprefinsql`
Skewed attributes	`orclskewedattribute`
Match DN	`orclMatchDnEnabled`

Restart the server after changing `orclskiprefinsql` or `orclskewedattribute`.

17.12.3 Modifying Attributes by Using ldapmodify

Most attributes can be modified by using the LDAP command `ldapmodify`.

You use a command line such as:

```
ldapmodify -D cn=orcladmin -q -p portNum -h hostname -f ldifFile
```

where *ldifFile* is an LDIF file.

17.12.3.1 Modifying Performance-Related Instance-Specific Configuration Entry Attributes

Here are some examples of LDIF files for modifying instance-specific configuration entry attributes.

orclgeneratechangelog

```
dn: cn=componentname,cn=osdldapd,cn=subconfigsubentry
changetype: modify
modify: orclgeneratechangelog
orclgeneratechangelog: 0
```

orclsizelimit

```
dn: cn=componentname,cn=osdldapd,cn=subconfigsubentry
changetype: modify
modify: orclsizelimit
orclsizelimit: 10000
```

orclenablegroupcache

```
dn: cn=componentname,cn=osdldapd,cn=subconfigsubentry
changetype: modify
modify: orclenablegroupcache
orclenablegroupcache: 0
```

17.12.3.2 Modifying Performance-Related Shared System Configuration Attributes in the DSA Configuration Entry

Here are some examples of LDIF files for modifying DSA configuration entry attributes.

orclskiprefinsql

```
dn: cn=dsaconfig,cn=configsets,cn=oracle internet directory
changetype: modify
replace: orclskiprefinsql
orclskiprefinsql: 1
```

orclinmemfiltprocess: One Filter is a Superset of Another

```
dn: cn=dsaconfig, cn=configsets, cn=oracle internet directory
changetype: modify
add: orclinmemfiltprocess
orclinmemfiltprocess: (objectclass=inetorgperson)(orclisenabled=TRUE)
```

orclskewedattribute

```
dn: cn=dsaconfig,cn=configsets,cn=oracle internet directory
changetype: modify
add: orclskewedattribute
orclskewedattribute: my_attribute
!
```

Restart the server after changing `orclskiprefinsql` or `orclskewedattribute`.

17.13 Setting Garbage Collection Configuration Attributes

The attributes `orclpurgetargetage` and `orclpurgeinterval` reside in the changelog purging configuration entry. You can change them with `ldapmodify` or Oracle Directory Services Manager.

17.13.1 Modifying Changelog Purging Attributes by Using ldapmodify

The following example is an LDIF file used to configure change log purging.

> **See Also:** "Change Log Purging" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* for a description of change log purging.

This example configures time-based purging for 120 hours (5 days). Use an LDIF file similar to this:

```
dn: cn=changelog purgeconfig,cn=purgeconfig,cn=subconfigsubentry
changetype:modify
replace: orclpurgetargetage
orclpurgetargetage: 240
```

To apply the LDIF file `mod.ldif`, type:

```
ldapmodify -D "cn=orcladmin" -q -p port -h host -D dn -q -f mod.ldif
```

> **See Also:** "Configuring Time-Based Change Log Purging" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

17.13.2 Modifying Changelog Purging in Oracle Directory Services Manager

You can modify `orclpurgetargetage` and `orclpurgeinterval` by using the data browser in Oracle Directory Services Manager. You cannot navigate to the changelog purging configuration entry directly in the data tree, but you can get to it by using an advanced search as follows:

1. On the Data Browser tab, click **Advanced**.
2. Expand **Garbage Collection** in the left pane, then select **changelog purgeconfig**. The Garbage Collector Window appears in the right pane.
3. In the right pane, enter the changes you want to make to the **Purge Target Age** and **Purge Interval.**
4. Choose **Apply**.

18

Oracle Virtual Directory Performance Tuning

This chapter provides tuning tips for Oracle Virtual Directory. It contains the following sections:

- Section 18.1, "About Oracle Virtual Directory"
- Section 18.2, "Basic Tuning Configurations"
- Section 18.3, "Additional Tuning Configurations"

18.1 About Oracle Virtual Directory

Oracle Virtual Directory is an LDAP Version 3-enabled service that provides an abstracted view of one or more enterprise data sources. Oracle Virtual Directory consolidates multiple data sources into a single directory view, enabling you to integrate LDAP-aware applications with diverse directory server data stores.

The information in this chapter assumes that you have reviewed the concepts and administration information in the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

Note: Oracle Virtual Directory's out of box configuration may not be optimal for many production and test deployments. You are encouraged to incorporate the recommendations listed in "Basic Tuning Configurations" to achieve optimal performance and availability.

18.2 Basic Tuning Configurations

The tuning configurations in this section apply to most deployments and usage scenarios. It is highly recommended that you review these configurations and implement those that are appropriate for your use case scenarios. The tuning information is summarized in Table 18–1.

- Tune the number of worker threads based on the number of central processing units (CPU) available for Oracle Virtual Directory Server on the system.

  The 'Threads' configuration parameter in the Oracle Virtual Directory Listener settings should be set to an appropriate value. The default out of box value for Threads in the Admin Gateway listener and DSML Gateway listener should be generally optimal and need not be changed. The number of Threads for the LDAP Listeners is typically the ones that need to be tuned since typically it is the LDAP Listeners that take on concurrent traffic from applications. A common

configuration is to have 10 threads per CPU. For example, if there are 4 central processing units on the system, then there would be 40 threads.

For more information, see "Managing Listeners" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

- Tune the Work Queue Capacity based on the expected maximum number of concurrent clients to a given LDAP Listener.

  The 'WorkQueueCapacity' configuration parameter in the Oracle Virtual Directory Listener settings should be set to an appropriate value. This ensures that the connection requests from LDAP clients are not rejected due to a lack of work queue capacity. Work elements are allocated on demand only and hence a value higher than actual estimate can be used.

  The Fusion Middleware Control Performance Monitor provides a historical report which contains the maximum number of connections. Use this report to determine how to adjust the connection value based on production data.

  For more information, see "Managing Listeners" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

- Tune the size of the LDAP connection pool in Oracle Virtual Directory LDAP Adapter to be at least as high as the total number of Threads configured in the Oracle Virtual Directory Listeners that actively use the LDAP Adapter.

  This ensures that in the worker threads have enough LDAP connections to process requests. The actual number of active adapters, active listeners and traffic pattern control the usage of connections. However, since connections that are idle in the LDAP Adapter connection pool are periodically closed, a higher value should not impact performance. Ensure that the back-end Directory Server is configured to handle the number of concurrent connections from Oracle Virtual Directory LDAP Adapter connection pool.

  For more information, see "Configuring LDAP Adapter" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

- Tune the maximum Java heap size of the JVM running Oracle Virtual Directory. This is to ensure that Oracle Virtual Directory has sufficient heap to handle the concurrent load.

  For more information, see "Controlling the Maximum Heap Size Allocated to the Oracle Virtual Directory Server" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

Table 18–1 Basic Tuning Configurations

Configuration Attribute	Category	Default Value	Recommended Value	Notes
Threads	Listener Properties	10	10 * Number Of central processing units (CPUs) available for Oracle Virtual Directory Server	Recommendation applies only to the active LDAP Listeners.
Work Queue Capacity	Listener Properties	1024	Expected Number of Max Concurrent Clients	Up to 2 GB on 32-bit systems and higher values on 64-bit systems.

Table 18–1 (Cont.) Basic Tuning Configurations

Configuration Attribute	Category	Default Value	Recommended Value	Notes
Max Pool Connections	LDAP Adapter Properties	10	Total Number of 'Threads parameter values for all active Listeners that use this Adapter	Ensure that the back-end Directory Servers can handle these connections.
Max Heap Size	System Properties	256 MB	Up to 2 GB on 32-bit systems and higher values on 64-bit systems.	Higher values protect against Out Of Memory errors. Ensure that there is sufficient RAM on the system to handle the configured value.

18.3 Additional Tuning Configurations

Depending on your Oracle Virtual Directory deployment's use case scenarios, the following tuning configurations may improve performance.

18.3.1 Database Adapters

The Database Adapter is a fully featured LDAP-to-JDBC gateway supporting translation of all LDAP operations (add, bind, delete, baseSearch, modify, wildCardSearch) into equivalent SQL prepared statement code. The Database Adapter uses JDBC class libraries to form connections to databases for the purpose of performing LDAP searches. The database libraries are generally provided by the database vendor.

> **Note:** For improved performance, tune the database before using the Database adapter. Consult your database documentation for more information. If the database being used is an Oracle database, see *Oracle Database Performance Tuning Guide*.

For optimal performance, consider the following configuration options for the database schema against which the Oracle Virtual Directory database adapter is configured:

- In general, the mapped columns in the underlying database schema should have an index defined if the mapped LDAP attribute is used in LDAP search filters.
- In scenarios where an LDAP attribute that is used in an LDAP search filter has a matching rule of 'caseIgnoreMatch', the mapped database table column for this attribute needs a function index to be defined for optimal look-up performance.

  For example, if LDAP attribute 'CN' is mapped to database schema column EMP.NAME, then a function index on UPPER(EMP.NAME) is required for optimal performance of LDAP search filters involving CN attribute.

  For more information on function-based indexes, see "Using Function-based Indexes for Performance" in *Oracle Database Performance Tuning Guide*.

Table 18–2 describes some additional Database Adapter settings:

Table 18–2 Database Adapter Settings

Parameter	Value	Notes
Adapter	Default: Active	An adapter can be configured as Active or Inactive. An inactive adapter can not start during a server restart or when you try to start it. The purpose of the Inactive setting is to keep old configurations available or on stand-by without having to delete them from the configuration.
Maximum Connections	Default: 10 connections	This defines the maximum connections the Database Adapter may make with the database.
Database Connection Timeout	Default: 10 seconds	The database connection timeout adapter property controls the LDAP request to wait for a connection to become available in the cache after reaching the maximum number of connections limit. If a connection does not become available within the number of seconds defined, the LDAP request fails. If database connection timeout system property is not used, the LDAP request waits 10 seconds for a connection to become available.

18.3.2 Join Adapters

If you are using Join Adapters, join only appropriate sources. For example if a deployment requires only to link attributes in the primary source under "cn=users" branch, create a primary adapter that only exposes this branch. And then create the join rule with that adapter. This can reduce the need for Oracle Virtual Directory to try to join entries that may never have corresponding linked entries.

> **Tip:** Always make sure that the attributes used by join rules are properly indexed.

18.3.3 General Filter Tuning

If a known client search filter does not apply to certain adapters, apply the filter to all applicable "Exclude Filters" to improve performance and reduce network traffic.

18.3.4 Load Balancer Local Store Adapter Tuning

Some load balancers query an LDAP server to determine if it is up or down. If your load balancer uses this feature - consider creating a local store adapter with a separate namespace (for example dc=loadbalancer) that is used only for the load balancer. While the performance impact of the load-balancer is probably not noticeable, by keeping it in a separate namespace. it makes it easier to exclude the load-balancer `KeepAlive` requests from creating large log files during troubleshooting.

18.3.5 Cache Plug-In Tuning

The CachePlug-in provides an in-memory cache for Oracle Virtual Directory. It has the ability to cache query results from any source for re-use by LDAP clients. This plug-in can improve performance for those applications where queries are highly repetitive.

To review cache operation and configuration, set VE logging level to 'Dump' to see more details. Because the cache is a normal plug-in, the cache can be configured to run

anywhere within Oracle Virtual Directory. It can be executed globally, or within the context of a single adapter. It can also be restricted to specific namespaces by using the namespace filtering available in standard plug-in configuration.

18.3.5.1 Cache Hit Logic

The cache works by storing query results and making them available for later use. If a query is repeated by the same user and the same attributes or a subset of attributes are requested, the cache can return its results instead of having Oracle Virtual Directory pull the information from the source. The plug-in can also be configured to allow cache hits to be shared between users.

Sharing cache entries between users should not be used unless the pass credentials are not being passed to back-end sources and Oracle Virtual Directory is solely responsible for security enforcement. Careful consideration should be given when sharing cache hits between users as it would then be possible for one user to see something they should not, since they may have access to a cache result from a more privileged user.

18.3.5.2 Cache Plug-in Memory Management

This plug-in periodically reviews the cache and checks for expired results, or entries that have been invalidated by a previous modify transaction. In the event that the cache quota is exceeded, the plug-in attempts to trim memory by purging the queries that were least recently used (LRU).

Table 18–3 describes some parameters used to tune the Memory Management Plug-in:

Table 18–3 Memory Management Plug-in Settings

Parameter	Value	Notes
Size	Default: 1000 entries	The maximum number of entries that may be cached at any one time.
MaxResultSize	Default: 1000 entries	The maximum number of entries that may be cached for any particular query.
Trimsize	Default: 1000 entries	When the maximum cache size is exceeded, the amount by which the cache manager must reduce the balance. Note: when necessary, trimming is done by purging expired queries first followed by queries in order of least recent use.
MaximumAge	Default: 600 seconds	The maximum age in seconds for any query/entry stored in the cache.
MaintenanceInterval	Default: 60 seconds	The interval in seconds between when the cache manager checks for expired queries.
BySubject	Default: 1 (not shared)	A flag (1 or 0) indicating whether cache results are shared between subjects. A value of 1 indicates that results are not be shared between subjects.

18.3.6 LDAP Listener Tuning

Table 18–4 describes some parameters used to tune the LDAP Listener:

Table 18–4 Listener Parameters

Parameter	Value	Notes
Backlog	Default: 128 requests	Specifies the maximum number of pending connection requests that are allowed to queue up before the server starts rejecting new connection attempts. The default value is sufficient in most cases and the need to change this value is very rare.
Reuse address	Default: False	This option determines whether LDAP listener should reuse socket descriptors. If enabled, the SO_REUSEADDR socket option is used on the Oracle Virtual Directory server listen socket to potentially allow the reuse of socket descriptors for clients in TIME_WAIT state.
Keep Alive	Default: False	This option determines whether the LDAP connection should use TCP keep-alive. If enabled, the SO_KEEPALIVE socket option is used to indicate that TCP keepalive messages should periodically be sent to the client to verify that the associated connection is still valid.
TCP No delay	Default: True	This option determines whether the LDAP connection should use TCP no-delay. If enabled, TCP_NODELAY socket option is used to ensure that response messages to the client are sent immediately rather than potentially waiting to determine whether additional response messages can be sent in the same packet.

Table 18–4 (Cont.) Listener Parameters

Parameter	Value	Notes
Read Timeout	Default: 0	This option enables/disables SO_ TIMEOUT with the specified timeout, in milliseconds. With this option set to a nonzero timeout, client connection to the Oracle Virtual Directory server can remain idle only for this amount of time. If the connection is idle for a period longer than the specified timeout, the client connection is terminated. A timeout of zero is interpreted as an infinite timeout. **Warning:** This option is equivalent to vde.soTimeoutFrontend system property in Oracle Virtual Directory version 10g. The vde.soTimeoutFrontend system property is not supported for 11g. Users must modify the value specified in system property The mapping of values from 10g to 11g are: .Enabled to 0 Disabled to nonzero amount of time in milliseconds

18.3.7 Server Tuning

Table 18–5 describes some basic parameters used to tune the server:

Table 18–5 Server Parameters

Parameter	Value	Notes
Anonymous Search Limit	Default: 1000	The maximum number of entries returned for an anonymous client.
Connection Timeout	Default: 120 (minutes)	The Connection Timeout system property is used to prevent service outages caused by clients that do not properly close connections. The value can be set in Oracle Enterprise Manager's Server Properties page. **Warning**: Setting to 0 disables the enforcement and client connections can not be closed regardless of how long they are inactive. The system property is not enforced on IP addresses and subjects that are exempt from the quota limit or that have disabled quota enforcement.

Table 18–5 (Cont.) Server Parameters

Parameter	Value	Notes
Logging Levels	Default: Error:1 (Severe)	By default, log messages are written to the access.log file only when logging is set to NOTIFICATION:1. To maintain performance, consider keeping the default log level or use WARNING:1 (WARNING) to limit the amount of information written to the access.log file.

19

Oracle Identity Federation Performance Tuning

Oracle Identity Federation is a standalone, self-contained federation server that enables single sign-on and authentication in a multiple-domain identity network. It contains the following sections:

- Section 19.1, "About Oracle Identity Federation"
- Section 19.2, "LDAP Tuning"
- Section 19.3, "Database Tuning"
- Section 19.4, "Oracle HTTP Server Tuning"
- Section 19.5, "SAML Protocol Tuning"

> **Note:** The configuration examples and recommended settings described in this chapter are for illustrative purposes only. Consult your own use case scenarios to determine which configuration options can provide performance improvements.

19.1 About Oracle Identity Federation

Oracle Identity Federation is a standalone, self-contained federation server that enables single sign-on (SSO) and authentication in a multiple-domain identity network. The federation single sign-on capabilities are based on the SAML 1.x/SAML 2.0/WS-Fed protocols. The server is a J2EE Application deployed in a WebLogic Managed Server. This enables users to federate in heterogeneous environments and business associations, whether they have implemented other Oracle Identity Management products in their solution set.

For more information see *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

19.2 LDAP Tuning

This section provides configuration settings that can be used to tune LDAP such as:

- Connection Pool Settings
- Connection Settings
- Federation Data Store Settings

For the best performance, review the tuning configurations in Chapter 2, "Top Performance Areas" before tuning Oracle Identity Federation.

19.2.1 Connection Pool Settings

When Oracle Identity Federation is integrated with LDAP Servers as a user data store, federation data store, or authentication engine, the server keeps a pool of LDAP connections that can be re-used for subsequent requests.

Oracle Identity Federation performs the following kind of operations to the LDAP Servers:

1. User Data Store
   - Locate users during assertion mappings
   - Retrieve attributes from the user record when creating an assertion
2. Authentication Engine
   - Locate user
   - Validate user credentials during authentication operations
3. Federation Data Store, if used
   - Create a federation record
   - Locate a federation record
   - Update or delete a federation record. The LDAP Connection Pool can be configured by:
     - Setting Maximum Connections to indicate how many LDAP connections can the pool contain.
     - Setting the Connection Wait Timeout which is the time that a thread waits before re-trying to get an LDAP connection when none are available in the pool and that the pool is at maximum capacity.

See "Configuring Oracle Identity Federation" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation* for more information on the User and Federation Stores as well as the LDAP Authentication Engine.

19.2.2 Connection Settings

When Oracle Identity Federation is integrated with LDAP Servers as a user data store, federation data store, or authentication engine, the LDAP run time connections can be configured. For more information, see "Configuring Oracle Identity Federation" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

The LDAP Connections can be configured by:

- Setting the LDAP Inactivity setting which tells Oracle Identity Federation how long an LDAP connection should be kept in a pool before being removed due to inactivity. Over time, the LDAP server may close some connections due to a long inactivity period, and if left unchecked, this can result in errors and may impact performance in Oracle Identity Federation.

  See "Configuring the LDAP Inactivity Setting" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

- Setting the LDAP Read Timeout Setting. Sometimes the LDAP server can become unresponsive, causing the thread/user to wait for a response or an error. To avoid

waiting too long for an error when the server is not responding, Oracle Identity Federation sets a read timeout property on the LDAP connection. If the LDAP server does not respond before the read timeout period, an error is generated. Oracle Identity Federation closes the connection, open a new one and re-issue the LDAP command.

See "Configuring the LDAP Read Timeout Setting" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

- Setting the High Availability (HA) LDAP Flag. When integrated with LDAP Servers that are deployed in HA mode, Oracle Identity Federation must configured to indicate that the LDAP Servers are in HA mode.

  See "Configuring High Availability LDAP Servers" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

19.2.3 Federation Data Store Settings

When using Oracle Internet Directory as the Federation Data Store, Oracle Identity Federation creates, locates, updates and deletes federation records containing Account Linking Information.

Oracle Identity Federation uses specific queries when interacting with Oracle Internet Directory, and the performance can be improved by creating filters in Oracle Internet Directory. If Oracle Internet Directory is used as the Federation Data Store, it is possible to tune the LDAP Server to improve the performance of the lookup operations. Oracle Identity Federation server can be configured to use a Federation Store to persist Federated Identities records.

The Federation server uses this store to:

- Lookup a federation record through different queries
- Create a federation record
- Delete a federation

In addition to the Oracle Identity Federation-related `orclinmemfiltprocess` filter `(objectclass=orclfeduserinfo)`, which is included by default, some Oracle Identity Federation environments might benefit from additional filters with the following formats:

`(orclfedserverid=`*`local_oif_server_id`*`)`

`(orclfedproviderid=`*`providerid_of_remote_server`*`)`

`(orclfedfederationtype=`*`n`*`)`

where `orclfedserverid` denotes the Oracle Identity Federation server that is making the query, `orclfedproviderid` is the identifier of a remote SAML server, and `orclfedfederationtype` is `1` or `3`. Use `1` as the value for `orclfedfederationtype` when Oracle Identity Federation is an Identity Provider and the remote provider is a Service Provider. Use `3` when Oracle Identity Federation is a Service Provider and the remote provider is an Identity Provider.

A deployment can be configured to work with many remote SAML servers, so there can be several `orclfedproviderid` filters and more than one `orclfedfederationtype` filter.

For example:

```
(orclfedserverid=my_oif_server)

(orclfedproviderid=http://server.example.com:7499/fed/idp)
```

```
(orclfedproviderid=http://server2.example.com:7492/fed/idp)
(orclfedfederationtype=1)
(orclfedfederationtype=3)
```

19.3 Database Tuning

This section provides configuration settings that can be used to tune the database.

See "Additional RDBMS Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

19.3.1 Data Sources

Oracle Identity Federation uses a J2EE data source to interact with a database for various operations, such as:

- Locating a user record in the User Data Store
- Retrieving attributes from a user record in the User Data Store
- Locating, creating, or deleting an Oracle Identity Federation record from the Federation Data Store
- Locating, creating, or deleting an Oracle Identity Federation transient record from the Session or Message Data Store. (A transient record can be a user session, an artifact record, or federation protocol or session state.)

When creating a data source in the WebLogic Administration Console that can be used by Oracle Identity Federation, the maximum and minimum connection settings should be tuned for better performance. Consult your use case scenarios to determine what the connections settings should be to improve performance in your application.

19.3.2 RDBMS Session Cache

When Oracle Identity Federation is integrated with RDBMS for its Session Data Store, the server uses a caching mechanism to improve performance at run time. This enables the server to keep a reference to recently used session objects in memory to avoid read access to the database.

To optimize RDBMS session caching, configure the following:

- Number of session objects kept in memory at a given time
- Length of time a specific session object is kept in memory

> **Note:** if Oracle Identity Federation is in High Availability (HA) mode with a load balancer, sticky sessions must be enabled to ensure that the cache is always reflecting accurate data.

See "Configuring RDBMS Session Cache" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

19.3.3 RDBMS Compression

To decrease the amount of data to be stored in an RDBMS, Oracle Identity Federation provides the capability to compress the data before storing it to the database.There are three kinds of data that can be compressed:

- `AuthnRequest` for SSO Artifact profile: when Oracle Identity Federation acts as an IdP for Liberty 1.x protocol, the server stores the `AuthnRequest` message in the RDBMS when the artifact profile is used.
- Assertion Response for SSO Artifact profile: when Oracle Identity Federation acts as an IdP for SSO protocols, the server stores the Response message containing the Assertion in the RDBMS when the artifact profile is used. This must be enabled if attributes are contained in the assertion.
- User Session Data: Oracle Identity Federation stores some session data related to the user at run time. If several attributes are stored in the User Session (set by a custom Authentication Engine, or because the Attributes Assertion storage was enabled when Oracle Identity Federation was a service provider), then compression should be used.

See "Configuring RDBMS Data Compression" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

19.4 Oracle HTTP Server Tuning

If Oracle Identity Federation is fronted by Oracle HTTP Server (OHS), then the configuration of the HTTP Server can be tuned to increase performance. For more information on Oracle HTTP Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

The following parameters can be changed in the httpd.conf file of the OHS. For additional Oracle HTTP tuning configurations, see Chapter 5, "Oracle HTTP Server Performance Tuning". Consult your use case scenarios to determine what your settings should be.

- Timeout
- KeepAlive
- MaxKeepAliveRequests
- KeepAlive TimeOut
- MinSpareServers
- MaxSpareServers
- StartServers
- MaxClients
- MaxRequestPerChild

After modifying these parameters, save and restart OHS.

19.5 SAML Protocol Tuning

The Security Assertion Markup Language (SAML) protocol involves interacting with remote servers through the use of the Simple Object Access Protocol (SOAP).

19.5.1 SOAP Connections

The Oracle Identity Federation server uses the SOAP protocol to send SAML Requests and to receive SAML Responses.

To optimize performance, configure the following SOAP connections:

- Total maximum number of SOAP connections that Oracle Identity Federation can open at the same time
- Maximum number of SOAP connections that Oracle Identity Federation can open at the same time to a given remote server

For more information, see "SOAP Binding" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

19.5.2 XML Digital Signatures

The SAML and WS-Fed protocols of Oracle Identity Federation rely on XML Digital Signatures to ensure the authenticity of messages and that messages are not tampered with.

When possible, sign the Assertion and/or the Response to prevent any modifications. When no XML Digital Signature is present on the message, the audited message that is archived does not contain any data that proves the authenticity and integrity of the message.

Configuring Oracle Identity Federation to *not* sign Assertion and/or Response may be appropriate if:

- Performance must be improved
- SSL with SSL authentication is enabled for SOAP communications
- Disabling XML Digital Signatures is compliant with company security regulations

---

Note: The content of the Assertion is viewable unless SAML 2 Encryption is used. Encrypting the Assertion is optional, but XML Encryption is resource intensive and decreases performance

---

19.5.3 POST and Artifact Single Sign-On Profiles

There are two Single Sign-On profiles defined by the SAML specifications:

- POST Profile

  In the POST profile, the Assertion transits through the user's browser, therefore the Assertion and/or the Response must be signed to ensure that the content has not been modified.

  Note: If the performance must be improved and if using the POST profile is compliant with company security regulations, then configuring Oracle Identity Federation to use the POST profile may be an option to improve performance.

- Artifact Profile

  In the Artifact profile, the Identity Provider creates a random identifier referencing the Assertion in the IdP's local store. (The Assertion is provided directly from the Identity Provider to the Service Provider.) That identifier is carried by the user's browser and presented to the Service Provider that contacts the Identity Provider to de-reference the identifier and retrieve the corresponding Assertion.

If the SOAP connection made from the SP to the IdP is encrypted using the SSL protocol with an SSL Server Certificate, then the SP authenticates the IdP and the content of the communication has not been tampered with: in this case, the transport layer is providing the authenticity and the integrity of the message, and the XML Digital Signature on the SAML Response and Assertion can be optional.

If no XML Digital Signature is present on the message, then the audited message that is archived does not contain any data that proves the authenticity and integrity of the message.

Note: Since the Artifact profile involves additional communication flow between the Service Provider and the Identity Provider, performance may be slower when using the Artifact profile.

20

Oracle Fusion Middleware Security Performance Tuning

Oracle Fusion Middleware security services enable you to secure critical applications and sensitive data. This chapter describes how you can configure security services for optimal performance.

This chapter contains the following topics:

- Section 20.1, "About Security Services"
- Section 20.2, "Detecting General Performance Issues"
- Section 20.3, "Oracle Platform Security Services Tuning"
- Section 20.4, "Oracle Web Services Security Tuning"

20.1 About Security Services

Oracle Fusion Middleware provides security services through Oracle Platform Security Services (OPSS) and Oracle Web Services.

- Oracle Platform Security Services

  Oracle Platform Services is a key component of Oracle Fusion Middleware. It offers an integrated suite of security services and is easily integrated with Java SE and Java EE applications that use the Java security model. Security Services includes features that implement user authentication, authorization, and delegation services that developers can integrate into their application environments. Instead of devoting resources to developing these services, application developers can focus on the presentation and business logic of their applications.

  Using Oracle Platform Security for Java, applications can enforce fine-grained access control upon resource users. The three key steps are:

  – Configure and invoke a login module, as appropriate. You can use provided login modules, or you can use custom login modules.

  – Authenticate the user attempting to log in, which is the role of the identity store service.

  – Authorize the user by checking permissions for any roles the user belongs to for whatever the user is attempting to accomplish, which is the role of the policy store service.

- Oracle Web Services Security

Oracle Web Services Security provides a framework of authorization and authentication for interacting with a web service using XML-based messages.

Note: The information in this chapter assumes that you have reviewed and understand the concepts and administration information for Oracle Fusion Middleware Security Services. For more information, see the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* before tuning any security parameters.

20.2 Detecting General Performance Issues

This section offers some general guidelines on how to identify a performance bottleneck and how to approach addressing such problems.

If you discover a performance bottleneck, you should first verify that you have addressed the expected traffic load throughout your Web services deployment. If there is a system in the critical path that is at 100% CPU usage, you may simply need to add one or more computers to the cluster.

If there is a bottleneck in your deployment, it is likely to be within one of the following:

- Traffic through a slow connection with an agent
- Latency in connections to third-party queuing systems like JMS

For any of these problems, check the following potential sources:

- Problems with policy assertions that include connections to outside resources, especially the following types:
  - Database Repositories
  - LDAP Repositories
  - Secured Resources
  - Proprietary Security Systems
- Problems with database performance

If you identify one of these as the cause of a bottleneck, you may need to change how you manage your database or LDAP connections or how you secure resources.

20.3 Oracle Platform Security Services Tuning

This section provides the following basic tuning configurations for Oracle Platform Security Services (OPSS):

- JVM Tuning Parameters
- LDAP Tuning Parameters
- Authentication Tuning Parameters
- Authorization Tuning Parameters
- OPSS Tuning Parameters for LDAP Policy Store

20.3.1 JVM Tuning Parameters

Tuning the JVM parameters can greatly improve performance. For example, the JVM Heap size should be tuned depending upon the number of roles and permissions in the store. At run time, all roles and permissions are stored in the in-memory cache. For more JVM tuning information, see Section 2.4, "Tune Java Virtual Machines (JVMs)".

20.3.2 LDAP Tuning Parameters

This section covers Lightweight Directory Access Protocol (LDAP) tuning. Oracle supports the management of policies in file-based repositories: Oracle Internet Directory and Oracle Virtual Directory.

If you encounter increased CPU usage due to high SQL execution times, see the following chapters for basic tuning configurations for large deployments:

- Oracle Internet Directory configuration settings can impact performance. For more information, see Chapter 17, "Oracle Internet Directory Performance Tuning".
- In addition to being configured as a LDAP server, Oracle Virtual Directory can also be configured as a local storage adapter (LSA). See Chapter 18, "Oracle Virtual Directory Performance Tuning".

20.3.3 Authentication Tuning Parameters

For OPSS Authentication tuning, see "Improving the Performance of WebLogic and LDAP Authentication Providers" in the Oracle Fusion Middleware Securing Oracle WebLogic Server guide at the Oracle Technology Network `http://download.oracle.com/docs/cd/E12840_01/wls/docs103/secmanage/atn.html#wp1199087`.

20.3.4 Authorization Tuning Parameters

The following parameters can be used to optimize Authorization:

Table 20–1 Authorization Parameters

Parameter	Value	Notes
`-Djps.combiner.optimize=true`	Default: True	This system property is used to cache the protection domains for a given subject.

Table 20–1 (Cont.) Authorization Parameters

Parameter	Value	Notes
`-Djps.combiner.optimize.lazyeval=true`	Default: True	This system property is used to evaluate a subject's protection domain when a checkpermission occurs.
`-Djps.policystore.hybrid.mode=false`	Default: False	This 'hybrid mode' property is used to facilitate transition from SUN RI of java.security.Policy to OPSS Java Policy Provider. The OPSS Java Policy Provider reads from both java.policy and system-jazn-data.xml. "Hybrid" mode can be disabled by setting system property "jps.policystore.hybrid.mode" to "false" when starting the WebLogic Server.
`-Djps.authz=ACC`	Default: ACC	Delegates the call to JDK API AccessController.checkPermission which reduces the performance impact at run time or while debugging.

20.3.5 OPSS Tuning Parameters for LDAP Policy Store

Table 20–2 provides OPSS Tuning parameters for LDAP Policy Store:

Table 20–2 OPSS Tuning Parameters for LDAP Policy Store

Parameter	Value	Notes
`oracle.security.jps.policystore.rolemember.cache.type`	Default: STATIC	Role Member Cache Type. Consider maintaining the default value for performance.
`oracle.security.jps.policystore.rolemember.cache.strategy`	Default: FIFO	Role Member Cache Strategy. Consider maintaining the default value for performance.
`oracle.security.jps.policystore.rolemember.cache.size`	Default: 1000	Role Member Cache Size. If performance is an issue, consider increasing the size to 5000 (if appropriate for your deployment).
`oracle.security.jps.policystore.policy.lazy.load.enable`	Default: TRUE	Enable Policy Lazy Load. Consider maintaining the default value for performance.
`oracle.security.jps.policystore.policy.cache.strategy`	Default: PERMISSION_FIFO	Permission Cache Strategy. Consider maintaining the default value for performance.
`oracle.security.jps.policystore.policy.cache.size`	Default: 1000	Permission Cache Size. If performance is an issue, consider increasing the size to 4000 (if appropriate for your deployment).
`oracle.security.jps.policystore.cache.updatable`	Default: TRUE	If TRUE, policy cache is incrementally updated for management operations. Consider maintaining the default value for performance.

Table 20–2 (Cont.) OPSS Tuning Parameters for LDAP Policy Store

Parameter	Value	Notes
`oracle.security.jps.policystore.refresh.enable`	Default: TRUE	This property is used for refresh enabling. Consider maintaining the default value for performance.
`oracle.security.jps.policystore.refresh.purge.timeout`	Default: 43200000	Forced policy store refresh time in milliseconds. Consider maintaining the default value for performance.
`oracle.security.jps.ldap.policystore.refresh.interval`	Default: 600000	Policy store refresh polling time in milliseconds for changes. Consider maintaining the default value for performance.

20.4 Oracle Web Services Security Tuning

Oracle Web Services Security provides a framework of authorization and authentication for interacting with a web service using XML-based messages. This section provides information on factors that might affect performance of the web service.

- Choosing the Right Policy
- Timestamp (On or Off)
- Policy Manager
- Configuring the Log Assertion to Record SOAP Messages
- Monitoring the Performance of Web Services

20.4.1 Choosing the Right Policy

Oracle Web Services Security supports many policies and the appropriate policies must be implemented based on the security need of the deployment. Careful consideration should be given to performance, since each additional policy can impact performance. For example Transport level security (SSL) is faster than Application level security, but transport level security can be vulnerable in multi-step transactions. Application level security has more performance implications, but provides end-to-end security.

See "Configuring Policies" in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* to determine which security policies are required for a deployment.

20.4.2 Timestamp (On or Off)

By default, the Oracle Web Service policies have timestamp turned `ON`. Consider the following when setting your timestamp: there may be security implications when the timestamp is `OFF` and there may be performance implications when the timestamp is `ON`. Review the implications of these settings before making a change.

Refer to the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* for more information.

20.4.3 Policy Manager

There is an inherent performance impact when using the database-based policy enforcement. When database policy enforcement is chosen, careful consideration must be given to the "polling" frequency of the agent to the database.

20.4.4 Configuring the Log Assertion to Record SOAP Messages

The request and response pipelines of the default policy include a log assertion that causes policy enforcement points (PEP) to record SOAP messages to either a database or a component-specific local file. There can be potential performance impacts to the logging level. To prevent performance issues, consider using the lowest logging level that is appropriate for your deployment.

The following logging levels can be configured in the log step:

- Header - Only the SOAP header is recorded.
- Body - Only the message content (body) is recorded.
- Envelope - The entire SOAP envelope, which includes both the header and the body, is recorded. Any attachments are not recorded.
- All - The full message is recorded. This includes the SOAP header, the body, and all attachments, which might be URLs existing outside the SOAP message itself.

Note: Typically, system performance improves when log files are located in topological proximity to the enforcement component. If possible, use multiple distributed logs in a highly distributed environment.

20.4.5 Monitoring the Performance of Web Services

You can monitor the performance on the following Oracle Web Services through the Web Services home page of Oracle Fusion Middleware Control:

- Endpoint Enabled Metrics such as:
  - Policy Reference Status
  - Total Violations
  - Security Violations
- Invocations Completed
- Response Time, in seconds
- Policy Violations such as:
  - Total Violations
  - Authentication Violations
  - Authorization Violations
  - Confidentiality Violations
  - Integrity Violations
- Total Faults

For general information on monitoring Oracle Fusion Middleware components, see Chapter 4, "Monitoring Oracle Fusion Middleware".

For detailed information on using Oracle Fusion Middleware Control to monitor Oracle Web Services, see "Monitoring the Performance of Web Services" in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

Part VI

Oracle WebCenter Suite Components

This part describes configuring Oracle WebCenter Suite components to improve performance. It contains the following chapter:

- Chapter 21, "Oracle WebCenter Performance Tuning"

21

Oracle WebCenter Performance Tuning

Oracle WebCenter Suite is the industry's only complete, open, and manageable portal platform that integrates Enterprise 2.0 capabilities into business processes and custom and packaged enterprise applications to create richer connections and deliver faster time-to-value. This section provides an overview of key Oracle WebCenter tuning concepts.

- Section 21.1, "About Oracle WebCenter"
- Section 21.3, "Tuning Environment Configurations"
- Section 21.2, "Tuning WebCenter Application Configuration"
- Section 21.4, "Tuning Back-End Component Configuration"
- Section 21.5, "Tuning the Database"

21.1 About Oracle WebCenter

Oracle WebCenter Suite 11*g* is an integrated suite of products used to create social applications, enterprise portals, communities, composite applications, and internet or intranet Web sites on a standards-based, service-oriented architecture (SOA). The suite combines the development of rich internet applications, a multi-channel portal framework, and a suite of horizontal Enterprise 2.0 applications, which provide content, presence, and social networking capabilities to create a highly interactive user experience. Interacting with services such as instant messaging, blogs, wikis, RSS, tags, Voice over IP, discussion forums, activities and social networks directly within the context of a portal or an application improves user and group productivity and enhances the return on IT investments.

Oracle WebCenter Spaces is an out-of-the-box WebCenter application that brings you the latest technology in terms of social networking, communication, collaboration, and personal productivity with no development effort. Through the robust set of integrated Web 2.0 services and applications provided by Oracle WebCenter Framework, Oracle WebCenter Composer and Business Dictionary, WebCenter Spaces enables you to deploy instant community portals, team sites and other collaborative applications. Oracle WebCenter Spaces provides three key capabilities within a single application: Personal Spaces, Business Role Pages and Group Spaces.

For more information about Oracle WebCenter, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* and *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*.

21.2 Tuning WebCenter Application Configuration

If WebCenter performance becomes an issue, consider tuning the following configuration parameters:

- HTTP Session Timeout
- JSP Page Timeout
- ADF Client State Token
- MDS Cache Size and Purge Rate
- Concurrency Management
- CRUD APIs (Create, Read, Update and Delete)

These parameters can be modified using configuration files for Oracle WebCenter applications. The main configuration files are adf-config.xml, connections.xml and web.xml. For details on locating these files in a WebCenter application deployment, as well as when to configure these files and which tools to use, see "Tuning WebCenter Application Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

21.3 Tuning Environment Configurations

This section describes provides information on configuring JDBC data source and memory size.

This section contains the following sections:

- JDBC Data Sources
- Adjust Memory Size

Note: The configuration examples and recommended settings described in this chapter are for illustrative purposes only. Consult your own use case scenarios to determine which configuration options can provide performance improvements.

21.3.1 JDBC Data Sources

WebCenter Spaces application uses two data sources: `mds-SpacesDS` and `WebCenterDS`.You can monitor you current JDBC connection usage in the WebLogic Server Administration Console. For more information, see "Configuring JDBC Data Sources" in *Oracle Fusion Middleware Configuring and Managing JDBC for Oracle WebLogic Server*.

21.3.2 Adjust Memory Size

If you have high loads on the system, for example you observe frequent garbage collection or 'out of memory' errors, you can increase the heap size and PermGen size as appropriate to your system's available physical memory.

See Tune Java Virtual Machines (JVMs) for more information on increasing heap size.

21.4 Tuning Back-End Component Configuration

Tuning information for back-end services, used by WebCenter applications, is documented in "Tuning Back-End Component Configuration" in *Oracle Fusion*

Middleware Administrator's Guide for Oracle WebCenter. Performance of back-end servers, for example, Worklists or Oracle Content Server, should be tuned as described in the related server documentation.

21.5 Tuning the Database

Tuning the database can greatly improve performance. Database tuning parameters are described in Section 2.6, "Tune Database Parameters". For more information on tuning the Oracle Database, see *Oracle Database Performance Tuning Guide*.

Part VII

Capacity Planning, Scalability, and Availability

This part describes how to plan your site for high traffic, scalability, and availability. It contains the following chapters:

- Chapter 22, "Capacity Planning"
- Chapter 23, "Using Clusters and High Availability Features"

22

Capacity Planning

Capacity Planning is the process of determining what type of hardware and software configuration is required to meet application needs. Like performance planning, capacity planning is an iterative process. A good capacity management plan is based on monitoring and measuring load data over time and implementing flexible solutions to handle variances without impacting performance.

> **Note:** The information contained in this chapter is meant to provide an overview of various techniques that can be used to develop an effective capacity management plan. The steps you take - and the plan you ultimately create - depends on your specific requirements and deployment structure.

The following sections provide an introduction to capacity planning:

- Section 22.1, "About Capacity Planning for Oracle Fusion Middleware"
- Section 22.2, "Determining Performance Goals and Objectives"
- Section 22.3, "Measuring Your Performance Metrics"
- Section 22.4, "Identifying Bottlenecks in Your System"
- Section 22.5, "Implementing a Capacity Management Plan"

22.1 About Capacity Planning for Oracle Fusion Middleware

While performance tuning can be defined as optimizing your *existing* system for better performance, capacity planning determines what your system needs (and when it needs it) to maintain performance in both steady-state and peak usage periods.

Capacity Planning involves designing your solution and testing the configuration, as well as identifying business expectations, periodic fluctuations in demand, and application constraints. You need to plan carefully, test methodically, and incorporate design principles that focus on performance. Before deploying any application into a production environment, the application should be put through a rigorous performance testing cycle. Creating an effective Capacity Management plan includes some of the same steps as performance planning:

- Step 1:Determining Performance Goals and Objectives
- Step 2: Measuring Your Performance Metrics
- Step 3: Identifying Bottlenecks in Your System
- Step 4: Implementing a Capacity Management Plan

22.1.1 Capacity Planning Factors to Consider

Before you can create a plan, you must have the data to support your deployment strategy. The following list of questions should be asked - and the information you receive should be analyzed carefully - to ensure a successful capacity management plan.

Table 22–1 Capacity Planning Factors to Consider

Capacity Planning Questions	For more information see,
What are your performance goals and objectives?	Section 22.2, "Determining Performance Goals and Objectives"
How many users need to run simultaneously (concurrently?)	Section 22.2, "Determining Performance Goals and Objectives"
Is the simulated workload adequate? (Is the workload likely to increase?)	Section 22.2, "Determining Performance Goals and Objectives"
Is the Oracle Fusion Middleware deployment configured to support clustering and other high availability factors?	Section 22.4.1, "Using Clustered Configurations"
Does the hardware meet the configuration requirements?	Section 22.5.1, "Hardware Configuration Requirements"
Do you have adequate JVMs to support your users?	Section 22.5.2, "JVM Requirements"
Is the database a limiting factor?	Section 22.5.4, "Database Configuration"

For more information, see Appendix B, "Related Reading and References".

22.2 Determining Performance Goals and Objectives

The first step in creating an effective capacity management plan is to determine your network load and performance objectives. You need to understand the applications deployed and the environmental constraints placed on the system. Ideally you have information about the levels of activity that components of the application are expected to meet, such as:

- The anticipated number of users.
- The number of concurrent sessions.
- The number of SSL connections required.
- The number and size of requests.
- The amount of data and its consistency.
- Determining your target CPU utilization.

Performance objectives are limited by constraints, such as

- The configuration of hardware and software such as CPU type, disk size versus disk speed, sufficient memory.
- The ability to interoperate between domains, use legacy systems, support legacy data.
- The security requirements and use of SSL. SSL involves intensive computing operations and supporting the cryptography operations in the SSL protocol can impact the performance of the WebLogic Server.

- Development, implementation, and maintenance costs.

  You can use this information to set realistic performance objectives for your application environment, such as response times, throughput, and load on specific hardware.

22.3 Measuring Your Performance Metrics

After you have determined your performance criteria in Section 22.2, "Determining Performance Goals and Objectives", take measurements of the metrics you can use to quantify your performance objectives. Benchmarking key performance indicators provides a performance baseline. See Chapter 4, "Monitoring Oracle Fusion Middleware" for information on measuring your performance metrics with Oracle Fusion Middleware applications.

22.4 Identifying Bottlenecks in Your System

Bottlenecks, or areas of marked performance degradation, should be addressed while developing your capacity management plan. If possible, profile your applications to pinpoint bottlenecks and improve application performance. Oracle provides the following profilers:

- Oracle Jrockit Mission Control provides profiling capabilities for processes using Jrockit JVM.

  `http://www.oracle.com/technology/products/jrockit/missioncontrol/index.html`

- Oracle Application Diagnostics provides profiling capabilities for java processing using SUN JDK.

  `http://www.oracle.com/technology/software/products/oem/htdocs/jade.html`

The objective of identifying bottlenecks is to meet your performance goals, not eliminate all bottlenecks. Resources within a system are finite. By definition, at least one resource (CPU, memory, or I/O) can be a bottleneck in the system. Planning for anticipated peak usage, for example, may help minimize the impact of bottlenecks on your performance objectives. See Appendix B, "Related Reading and References".

There are several ways to address system bottlenecks. Some common solutions include:

- Using Clustered Configurations
- Using Connection Pooling
- Setting the Max HeapSize on JVM
- Increasing Memory or CPU
- Segregation of Network Traffic
- Segregation of Processes and Hardware Interrupt Handlers

22.4.1 Using Clustered Configurations

Clustered configurations distribute work loads among multiple identical cluster member instances. This effectively multiplies the amount of resources available to the distributed process, and provides for seamless fail over for high availability.

For more information see Chapter 23, "Using Clusters and High Availability Features".

22.4.2 Using Connection Pooling

You may be able to improve performance by using existing database connections. You can limit the number of connections, timing of the sessions and other parameters by modifying the connection strings.

See Section 2.7, "Reuse Database Connections" for more information on configuring the database connection pools.

22.4.3 Setting the Max HeapSize on JVM

This is a application-specific tunable that enables a trade off between garbage collection times and the number of JVMs that can be run on the same hardware. Large heaps are used more efficiently and often result in fewer garbage collections. More JVM processes offer more fail over points.

See Section 2.4, "Tune Java Virtual Machines (JVMs)" for more information.

22.4.4 Increasing Memory or CPU

Aggregating more memory and/or CPU on a single hardware resource allows localized communication between the instances sharing the same hardware. More physical memory and processing power on a single machine enables the JVMs to scale and run much larger and more powerful instances, especially 64-bit JVMs. Large JVMs tend to use the memory more efficiently, and Garbage Collections tend to occur less frequently. In some cases, adding more CPU means that the machine can have more instruction and data cache available to the processing units, which means even higher processing efficiency.

See Section 2.2, "Ensure the Hardware Resources are Sufficient" for more information.

22.4.5 Segregation of Network Traffic

Network-intensive applications can introduce significant performance issues for other applications using network. Segregating the network traffic of time-critical applications from network-intensive applications, so that they get routed to different network interfaces, may reduce performance impacts. It is also possible to assign different routing priorities to the traffic originating from different network interfaces.

22.4.6 Segregation of Processes and Hardware Interrupt Handlers

When planning for the capacity that a specific hardware resource can handle, it is important to understand that the operating system may not be able to efficiently schedule the JVM processes as well as other system processes and hardware interrupt handlers. The JVM may experience performance impacts if it shares even a few of its CPU cores with the hardware interrupt handlers. For example, disk and network-intensive applications may induce performance impacts that are disproportionate to the load experienced by the CPU. In addition, hardware interrupts can prevent the active Java threads from reaching a "GC-safe point" efficiently. Separating frequent hardware interrupt handlers from the CPUs running the JVM process can reduce the wait for Garbage Collections to start.

It may also be beneficial to dedicate sibling CPUs on a multi-core machine to a single JVM to increase the efficiency of its CPU cache. If multiple processes have to share the CPU, the data and instruction cache can be contaminated with the data and instructions from both processes, thus reducing the amount of the cache used effectively. Assigning the processes to specific CPU cores, however, can make it impossible to use other CPU cores during peak load bursts. The capacity management

plan should include a determination on whether the CPUs should be used more efficiently for the nominal load, or should there be some extra capacity for a burst of activity.

22.5 Implementing a Capacity Management Plan

Once you have defined your performance objectives, measured your workload, and identified any bottlenecks, you must create and implement a capacity management plan. The goal of your plan should be to meet or exceed your performance objectives (especially during peak usage periods) and to allow for future workload increases. To achieve your performance objectives, you must implement your management plan and then continuously monitor the performance metrics as discussed in Chapter 4, "Monitoring Oracle Fusion Middleware".

Since no two deployments are identical, its virtually impossible to illustrate how a capacity management plan would be implemented for all configurations. Capacity planning is an iterative process and your plan must be calibrated as changes in your workload or environment change. The following section provides key factors that should be addressed in the plan:

22.5.1 Hardware Configuration Requirements

There is no single formula for determining your hardware requirements. The process of determining what type of hardware and software configuration involves assessment of your system performance goals and an understanding of your application. Capacity planning for server hardware should focus on maximum performance requirements.

The hardware requirements you have today are likely to change. Your plan should allow for workload increases, environment changes (such as added servers or 3rd party services), software upgrades (operating systems, middleware or other applications), network connectivity and network protocols.

22.5.1.1 CPU Requirements

Your target CPU usage should not be 100%, you should determine a target CPU utilization based on your application needs, including CPU cycles for peak usage. If your CPU utilization is optimized at 100% during normal load hours, you have no capacity to handle a peak load. In applications that are latency sensitive and maintaining the ability for a fast response time is important, high CPU usage (approaching 100% utilization) can reduce response times while throughput stays constant or even increases because of work queuing up in the server. For such applications, a 70% - 80% CPU utilization recommended. A good target for non-latency sensitive applications is about 90%.

22.5.1.2 Memory Requirements

Memory requirements are determined by the optimal heap size for the applications you are going to use, for each JVM co-located on the same hardware. Each JVM needs up to 500MB in addition to the optimal heap size; the actual impact to performance depends on the JVM brand, and on the type of application being run. For example, applications with more Java classes loaded need more space for compiled classes. 32-bit JVMs normally cannot exceed a limit of approximately 3GB on some architecture when a limit is imposed by the hardware architecture and the Operating System. It is recommended to reserve some memory for the Operating System, IO buffers and shared-memory devices.

22.5.2 JVM Requirements

The number of users/processes that a single Java Virtual Machine (JVM) can handle varies widely on the types of requests and the type of JVM you are running. As part of your performance monitoring and benchmarking procedures, you should determine how many and what kinds of processes are executed and determine if your hardware meets the requirements for your specific JVM.

22.5.3 Managed Servers

Using multiple managed servers across multiple nodes in a clustered configuration is recommended for both high performance and reliability. It is important to note, however, that having multiple managed servers may mean using more memory which can enable some applications to optimize certain operations in-memory, therefore reducing impact of disk, database and network latency.

For more information on using clustered configurations, see "Understanding Managed Servers and Managed Server Clusters" in *Oracle Fusion Middleware Administrator's Guide*.

22.5.4 Database Configuration

To maintain sustained performance, you must ensure that your existing database can scale with the increases in capacity planned for the application server tier. Tuning the database parameters and monitoring database metrics during peak usage, can help you determine if the existing database resources can scale to handle increased loads. You may need to add additional memory or upgrade the database hardware configuration. For more information on tuning an Oracle database, see the *Oracle Database Performance Tuning Guide*.

In some cases, however, you may find that the database is still not able to effectively manage increases in load, even after increasing the memory or upgrading the CPU. In these situations, consider deploying an Oracle Real Application Cluster (RAC) environment to handle the increases. Oracle RAC configurations not only provide enhanced performance, but they can also improve reliability and scalability. For more information on Oracle RAC, see *Oracle Real Application Clusters Administration and Deployment Guide*.

23

Using Clusters and High Availability Features

A high availability architecture is one of the key requirements for any Enterprise Deployment. Oracle Fusion Middleware has an extensive set of high availability features, which protect its components and applications from unplanned down time and minimize planned downtime.

This chapter provides an overview of the architecture, interaction, and dependencies of Oracle Fusion Middleware components, and explains how they can be deployed in a high availability architecture to maximize performance.

This chapter includes the following sections:

- Section 23.1, "About Clusters and High Availability Features"
- Section 23.2, "Using Clusters with Oracle Fusion Middleware"
- Section 23.3, "Using High Availability Features with Oracle Fusion Middleware"

> **Note:** Using clusters and other high availability options is a complex and detailed process. This chapter is meant to introduce the concepts as they relate to Oracle Fusion Middleware. Table 23–1 provides a list of Oracle Fusion Middleware guides that contain detailed high availability information.

23.1 About Clusters and High Availability Features

One of the most important factors in both high availability and performance is the use of **clusters**. A cluster is a set of processes running on single or multiple computers that share the same workload. Using a clustered configuration promotes scalability, high availability, and performance.

High availability refers to the ability of users to access a system without loss of service. Deploying a high availability system minimizes the time when the system is down, or unavailable and maximizes the time when it is running, or available. See

Details about using clusters and other high availability features can be located in the application-specific guides listed in Table 23–1:

Table 23–1 Clusters and High Availability Information in Oracle Fusion Middleware Documentation

Component	Location of Information
Oracle Fusion Middleware	*Oracle Fusion Middleware Administrator's Guide*
Oracle WebLogic Server	*Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server* *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
Oracle SOA Suite	The *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite* The *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite*
Oracle WebCenter	The *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* The *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle WebCenter*
Oracle ADF	The *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework* The *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*
Oracle Fusion Middleware Backup and Recovery	The *Oracle Fusion Middleware Administrator's Guide*
Oracle Web Cache	The *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*
Oracle Identity Management	The *Oracle Fusion Middleware Installation Guide for Oracle Identity Management* The *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*
Oracle Virtual Directory	The *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*
Oracle HTTP Server	The *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*
Oracle Internet Directory	The *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
Oracle Repository Creation Utility (RCU)	The *Oracle Fusion Middleware Repository Creation Utility User's Guide*
Oracle Portal	The *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*

23.2 Using Clusters with Oracle Fusion Middleware

For production environments that require increased application performance, throughput, or high availability, you can configure two or more Managed Servers to operate as a cluster. A cluster is a collection of multiple Oracle WebLogic Server server instances running simultaneously and working together to provide increased scalability and reliability.

For more information on using clusters with Oracle Fusion Middleware, see the following:

- "Understanding Managed Servers and Managed Server Clusters" in *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*
- *Oracle Real Application Clusters Administration and Deployment Guide*

23.3 Using High Availability Features with Oracle Fusion Middleware

In addition to using a clustered architecture within your Fusion Middleware components, there are a number of high availability features built-in to ensure your applications are continuously accessible by the users. The following list provides a few options for setting up a comprehensive high availability system. The options that you integrate depend on your overall performance goals as well as your system architecture. This list is meant to provide examples only.

- **Process death detection and automatic restart**

  Processes may die unexpectedly due to configuration or software problems. A proper process monitoring and restart system should constantly check the health of the applications and restart them when problems appear.

  A system process should also maintain the number of restarts within a specified time interval. This is also important since continually restarting within short time periods may lead to additional faults or failures. Therefore a maximum number of restarts or retries within a specified time interval should also be designed as well.

- **State replication and routing**

  For stateful applications, client state can be replicated to enable stateful failover of requests in the event that processes servicing these requests fail.

- **Failover**

  With a load-balancing mechanism in place, the instances are redundant. If any of the instances fail, requests to the failed instance can be sent to the surviving instances.

- **Server load balancing**

  When multiple instances of identical server components are available, client requests to these components can be load balanced to ensure that the instances have roughly the same workload.

- **Disaster Recovery**

  Disaster recovery solutions typically set up two homogeneous sites, one active and one passive. Each site is a self-contained system. The active site is generally called the production site, and the passive site is called the standby site. During normal operation, the production site services requests; in the event of a site failover or switchover, the standby site takes over the production role and all requests are routed to that site. To maintain the standby site for failover, not only must the standby site contain homogeneous installations and applications, data and configurations must also be synchronized constantly from the production site to the standby site.

For more information see the *Oracle Fusion Middleware High Availability Guide*.

Part VIII

Appendixes

This part contains the following appendixes:

- Appendix A, "Instrumenting Applications with DMS"
- Appendix B, "Related Reading and References"

A

Instrumenting Applications with DMS

Oracle Dynamic Monitoring Service (DMS) enables application developers, support analysts, system administrators, and others to measure application-specific performance information. This appendix describes DMS and shows a sample application that demonstrates how to use DMS to instrument Java applications.

> **Note:** Oracle Fusion Middleware provides several built-in metrics. Using DMS to instrument applications adds new metrics to the set of built-in metrics.

This appendix includes the following sections:

- Section A.1, "About DMS Performance Metrics"
- Section A.2, "Adding DMS Instrumentation to Java Applications"
- Section A.3, "Validating and Testing Applications Using DMS Metrics"
- Section A.4, "Understanding DMS Security Considerations"
- Section A.5, "Conditional Instrumentation Using DMS Sensor Weight"
- Section A.6, "Dumping DMS Metrics to Files"
- Section A.7, "Resetting and Destroying Sensors"
- Section A.8, "DMS Coding Recommendations"
- Section A.9, "Using a High Resolution Clock to Increase DMS Precision"
- Section A.10, "Rolling Up DMS Data for Descendent Nouns"

A.1 About DMS Performance Metrics

The Dynamic Monitoring Service (DMS) API enables you to add performance instrumentation to Oracle Fusion Middleware applications. At run time Oracle Fusion Middleware collects performance information, called DMS metrics, that developers, system administrators, and support analysts use to help analyze system performance or monitor system status.

This section includes the following subsections:

- Section A.1.1, "Instrumenting Applications with DMS"
- Section A.1.2, "Monitoring DMS Metrics"
- Section A.1.3, "Understanding DMS Terminology (Nouns and Sensors)"
- Section A.1.4, "DMS Naming Conventions"

Oracle Fusion Middleware components provide several predefined metrics. Use the Fusion Middleware Control online help to obtain a definition of a specific performance metric. See Section 4.2.1, "Viewing Performance Metrics Using Fusion Middleware Control" for information on accessing metric information.

A.1.1 Instrumenting Applications with DMS

DMS **Instrumentation** refers to the process you use when you insert DMS calls into application code. By using the DMS API, you can collect and analyze performance information for your application.

To create DMS metrics you add DMS API calls that notify DMS when events occur, when important intervals begin and end, or when pre-computed values change their state. At run time, DMS stores metrics in memory and enables you to save or view the metrics.

Note: Ensure that the `dms.jar` file is in your application classpath.

By default, the `dms.jar` file is located in the following: <ORACLE_HOME>/modules/oracle.dms_11.1.1/dms.jar.

Oracle Fusion Middleware includes built-in DMS metrics. By adding DMS calls to your applications you expand the set of built-in metrics. When you instrument your applications with DMS calls, you use the same API that the built-in metrics use. In addition, to save and display your metrics, you use the same monitoring tools that you use with built-in metrics.

Note that oracle.dms.trace.triggered.enable defaults to TRUE in Oracle Fusion Middleware 11*g* release 1 (11.1.1) and defaults to FALSE in 11*g* release 1 (11.1.1.2.0).

See Section A.2, "Adding DMS Instrumentation to Java Applications" for details.

A.1.2 Monitoring DMS Metrics

Monitoring DMS metrics refers to the process of retrieving performance metrics. When an application runs, DMS stores metrics in memory; you can view the metrics on the console or using a web browser.

Oracle Fusion Middleware provides several run time tools for viewing and saving DMS metrics, including the WLST command (with the `displayMetricTableNames`, `displayMetricTables`, and `dumpMetrics` subcommands) and the Spy servlet.

Note: To use WLST commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide* for more information.

Example A–1 shows the output of `wlst dumpMetrics`.

Example A–1 Sample Output of wlst dumpMetrics for the dmsDemo Application

```
/dmsDemo [type=n/a]
 /dmsDemo/BasicBinomial [type=MathSeries]
  computeSeries.active: n threads
  computeSeries.avg: n msecs
  computeSeries.completed: n ops
```

```
computeSeries.maxActive: n threads
computeSeries.maxTime: n msecs
computeSeries.minTime: n msecs
computeSeries.time: n msecs
lastComputed.value: n
loops.count: n ops
```

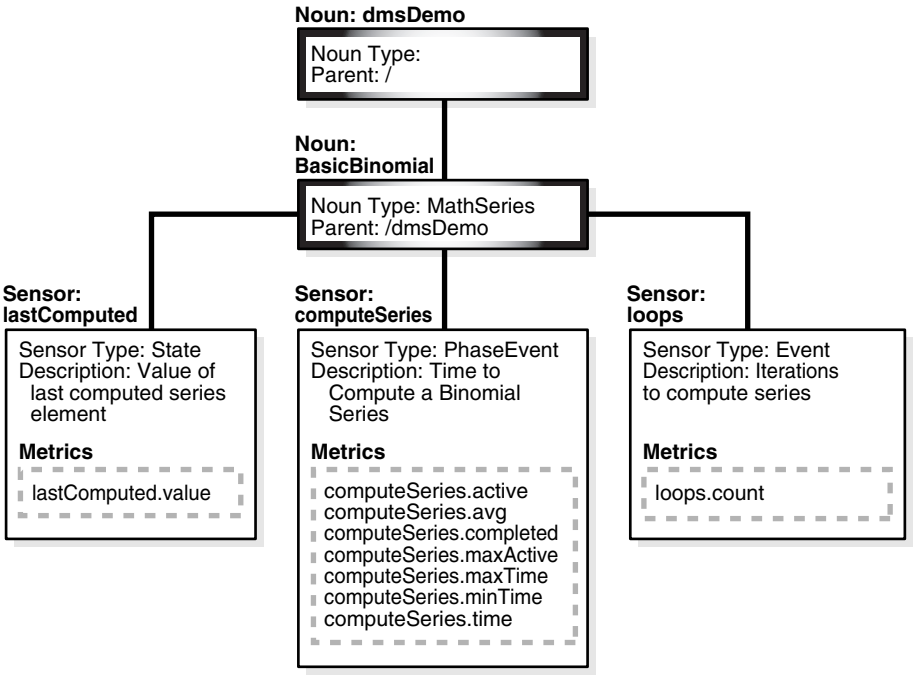
A.1.3 Understanding DMS Terminology (Nouns and Sensors)

This section introduces terminology related to DMS. This section contains the following subsections:

- Section A.1.3.1, "DMS Metrics"
- Section A.1.3.2, "DMS Sensors"
- Section A.1.3.3, "DMS Nouns"
- Section A.1.3.4, "DMS Roll-up Nouns"
- Section A.1.3.5, "DMS Object Relationships"

Figure A–1 illustrates the organization of a set of DMS metrics that correspond to the metrics in the demo application described in this chapter and the metrics shown in Example A–1.

Figure A–1 Organization of Sample Metrics for dmsDemo Application



A.1.3.1 DMS Metrics

DMS metrics provide performance information that developers, system administrators, and support analysts can use to help analyze system performance or monitor system status.

A.1.3.2 DMS Sensors

DMS **Sensors** measure performance data and enable DMS to define and collect a set of metrics. Certain metrics are always included with a Sensor and other metrics are optionally included with a Sensor.

DMS has three different kinds of sensors:

- Section A.1.3.2.1, "DMS PhaseEvent Sensors"
- Section A.1.3.2.2, "DMS Event Sensors"
- Section A.1.3.2.3, "DMS State Sensors"

A.1.3.2.1 DMS PhaseEvent Sensors A DMS **PhaseEvent Sensor** measures the time spent in a specific section of code that has a beginning and an end. Use a PhaseEvent Sensor to track time in a method or in a block of code.

DMS can calculate optional metrics associated with a PhaseEvent, including the average, maximum, and minimum time that is spent in the PhaseEvent Sensor.

Table A–1 lists the metrics available with PhaseEvent Sensors.

Table A–1 DMS PhaseEvent Sensor Metrics

Metric	Description
sensor_name.time	Specifies the total time spent in the phase *sensor_name*. Default metric: time is a default PhaseEvent Sensor metric.
sensor_name.completed	Specifies the number of times the phase *sensor_name* has completed since the process was started. Optional metric
sensor_name.minTime	Specifies the minimum time spent in the phase *sensor_name*, for all the times the *sensor_name* phase completed. Optional metric
sensor_name.maxTime	Specifies the maximum time spent in the phase *sensor_name*, for all the times the *sensor_name* phase completed. Optional metric
sensor_name.avg	Specifies the average time spent in the phase *sensor_name*, computed as the (total time)/(number of times the phase completed). Optional metric
sensor_name.active	Specifies the number of threads in the phase *sensor_name*, at the time the DMS statistics are gathered (the value may change over time). Optional metric
sensor_name.maxActive	Specifies the maximum number of concurrent threads in the phase *sensor_name*, since the process started. Optional metric

A.1.3.2.2 DMS Event Sensors A DMS **Event Sensor** counts system events. Use a DMS Event Sensor to track system events that have a short duration, or where the duration of the event is not of interest but the occurrence of the event is of interest.

Table A–2 describes the metric that is associated with an Event Sensor.

Table A–2 DMS Event Sensor Metrics

Metric	Description
sensor_name.count	Specifies the number of times the event has occurred since the process started, where *sensor_name* is the name of the Event Sensor as specified in the DMS instrumentation API. Default: count is the default metric for an Event Sensor. No other metrics are available for an Event Sensor.

A.1.3.2.3 DMS State Sensors A DMS **State Sensor** tracks the value of Java primitives or the content of a Java object. Supported types include integer, double, long, and object. Use a State Sensor when you want to track system status information or when you need a performance metric that is not associated with an event. For example, use State Sensors to track queue lengths, pool sizes, buffer sizes, or host names. You assign a precomputed value to a State Sensor.

Table A–3 describes the State Sensor metrics. State Sensors support a default metric value, as well as optional metrics. The optional minValue and maxValue metrics only apply for State Sensors if the State Sensor represents a numeric Java primitive (of type integer, double, or long).

Table A–3 DMS State Sensor Metrics

Metric	Description
sensor_name.value	Specifies the metric value for *sensor_name*, using the type assigned when *sensor_name* is created. Default: value is the default State metric.
sensor_name.count	Specifies the number of times *sensor_name* is updated. Optional metric
sensor_name.minValue	Specifies the minimum value for *sensor_name* since startup. Optional metric
sensor_name.maxValue	Specifies the maximum value this *sensor_name* since startup. Optional metric

A.1.3.3 DMS Nouns

DMS **Nouns** organize performance data. Sensors, with their associated metrics, are organized in an hierarchy according to Nouns. Nouns enable you to organize DMS metrics in a manner comparable to a directory structure in a file system. For example, Nouns can represent classes, methods, objects, queues, connections, applications, databases, or other objects that you want to measure.

A **Noun type** is a name that reflects the set of metrics being collected.

The Noun naming scheme uses a '/' as the root of the hierarchy, with each Noun acting as a container under the root, or under its parent Noun.

A.1.3.4 DMS Roll-up Nouns

DMS Roll-up Nouns are nouns that DMS generates when you include instrumentation to request a set of aggregate nouns. The roll-up noun contains metrics from a set of Sensors in the descendent nouns of a specified noun type. A roll-up noun also contains summary information.

See Also: Section A.10, "Rolling Up DMS Data for Descendent Nouns"

A.1.3.5 DMS Object Relationships

This section describes the object relationships and attributes for DMS metrics, Sensors, and Nouns.

Table A–4 describes the relationships between DMS objects. Figure A–1 illustrates the relationships shown in Table A–4 using a sample set of metrics.

Table A–4 DMS Object Relationships and Attributes

Object	Contains	Attributes
Noun	Sensors or other Nouns	Name, Noun Type, Parent
Sensor There are three Sensor Types: ■ PhaseEvent ■ Event ■ State	Metrics	Name, Description, Parent
Metric	Value	Name, Units designation

A.1.4 DMS Naming Conventions

When you create DMS nouns and sensors, you should follow the guidelines in this section so that users can easily understand DMS metrics across applications and across Oracle Fusion Middleware components.

Note: View the naming conventions as guidelines; for each convention there may be an exception. Try to be as clear as possible; if there is a conflict, you may need to make an exception.

This section includes the following subsections:

- Section A.1.4.1, "General DMS Naming"
- Section A.1.4.2, "General DMS Naming Conventions and Character Sets"
- Section A.1.4.3, "Noun and Noun Type Naming Conventions"
- Section A.1.4.4, "Sensor Naming Conventions"

A.1.4.1 General DMS Naming

A Noun name is a simple string, not including a delimiter. For example, `BasicBinomial` is a Noun name. A Noun full name consists of the Noun name, preceded by the full name of its parent, and a delimiter. For example `/dmsDemo/BasicBinomial` is a Noun full name.

A Sensor name is a simple string, not including the "`.`" or the derivation. For example, `computeSeries`, `loops`, and `lastComputed` are Sensor names.

A Sensor full name consists of the Sensor name, preceded by the name of its associated Noun, and a delimiter. Examples: `/dmsDemo/BasicBinomial/computeSeries`, `/dmsDemo/BasicBinomial/loops`, `/dmsDemo/BasicBinomial/lastComputed`.

A DMS metric name consists of a Sensor name plus the "." character plus the metric. For example, `computeSeries.time`, `loops.count`, and `lastComputed.value` are valid DMS metric names.

> **Note:** The suffixes .time, .count, and .value are immutable. Sensor and Noun names, however, can be modified as needed.

A.1.4.2 General DMS Naming Conventions and Character Sets

DMS names should be as compact as possible. Whenever possible, when you define Noun and Sensor names, avoid special characters such as white space, slashes, periods, parenthesis, commas, and control characters.

Table A–5 shows DMS replacement for special characters in names.

Table A–5 Replacement for Special Characters in DMS Names

Character	DMS Replacement Character
Space character	Underscore character: _
Period character: .	Underscore character: _
Control character	Underscore character: _
Less than character: <	Open parenthesis: (
Greater than character: >	Close parenthesis:)
Ampersand: &	Caret: ^
Double quote: "	Backquote: `
Single quote: '	Backquote: `

> **Note:** Oracle Fusion Middleware includes several built-in metrics. The Oracle Fusion Middleware built-in metrics do not always follow the DMS naming conventions.

A.1.4.3 Noun and Noun Type Naming Conventions

A Noun name should identify a specific entity of interest.

Noun types should have names that clearly reflect the set of metrics being collected. For example, Servlet is the type for a Noun under which the metrics that are specific to a given servlet fall.

Noun type names should start with a capital letter to distinguish them from other DMS names. All Nouns of a given type should contain the same set of sensors.

A.1.4.4 Sensor Naming Conventions

The following list outlines DMS Sensor naming conventions:

- Sensor names should be descriptive, but not redundant. Sensor names should not contain any part of the Noun name hierarchy, or type, as this is redundant.
- Sensor names should avoid containing the value for the individual metrics.
- Where multiple words are required to describe a Sensor, the first word should start with a lowercase letter, and the following words should start with uppercase letters. Example: `computeSeries`

- In general, avoid using a "/" character in a Sensor name. However, there are cases where it makes sense to use a name that contains "/" . If a "/" is used in a Noun or Sensor name, then when you use the Sensor in a string with DMS methods, you need to use an alternative delimiter, such as "," or "_", which does not appear anywhere in the path; this enables the "/" to be properly understood as part of the Noun or Sensor name rather than as a delimiter.

  For example, a child Noun can have a name such as:

  ```
  examples/jsp/num/numguess.jsp
  ```

  and you can look this up using the string:

  ```
  ,default,WEBs,defaultWebApp,JSPs,example/jsp/num/numguess.jsp,service
  ```

  where the delimiter is the "," character.

- Event Sensor and PhaseEvent Sensor names should have the form *verbNoun*. Examples: `activateInstance` and `runMethod`. When a PhaseEvent monitors a function, method, or code block, it should be named to reflect the task performed as clearly as possible.
- The name of a State Sensor should be a noun, possibly preceded by an adjective, which describes the semantics of the value which is tracked with this State Sensor. Examples: `lastComputed`, `totalMemory`, `port`, `availableThreads`, `activeInstances`
- To avoid confusion, do not name Sensors with strings such as ".time", ".value", or ".avg", which are names of Sensor metrics, as shown in Table A–1, Table A–2, and Table A–3.

A.2 Adding DMS Instrumentation to Java Applications

You can collect performance information for Java applications by adding DMS instrumentation to existing applications or by creating new applications that include DMS instrumentation.

You can download the DMS samples in this chapter from Oracle Technology Network:

`http://www.oracle.com/technology/tech/java/oc4j/demos/904/DMS/dmsDemo.zip`

The `dmsDemo.zip` file includes a ready-to-deploy .ear file and source code with build instructions. The demo includes two servlets: `BasicBinomial.java` and `ImprovedBinomial.java`.

The BasicBinomial servlet shows how to use the DMS API to add DMS Nouns and Sensors.

The ImprovedBinomial servlet expands on the BasicBinomial and illustrates measuring the improved code, as compared with the BasicBinomial. ImprovedBinomial servlet also shows how to add more costly metrics that gather more detailed information.

Refer to the sample code for full details on the examples in this chapter.

To use DMS instrumentation, add DMS calls by performing the following steps:

- Section A.2.1, "Including DMS Imports"
- Section A.2.2, "Organizing Performance Data"
- Section A.2.3, "Defining and Using Metrics for Timing"

- Section A.2.4, "Defining and Using Metrics for Counting"
- Section A.2.5, "Defining and Using Metrics for Recording Status Information (State Sensors)"

A.2.1 Including DMS Imports

To use DMS you need to add DMS imports. The following example shows the imports that the sample application `BasicBinomial.java` requires.

```
import oracle.dms.instrument.DMSConsole;
import oracle.dms.instrument.Event;
import oracle.dms.instrument.Noun;
import oracle.dms.instrument.PhaseEvent;
import oracle.dms.instrument.State;
import oracle.dms.instrument.Sensor;
```

A.2.2 Organizing Performance Data

In the sample application with the BasicBinomial and ImprovedBinomial servlets, the Nouns are organized as follows:

Example A–2 shows a section of code from `BasicBinomial.java`. It shows calling `Noun.create` to create the `/dmsDemo` noun root and the `BasicBinomial` noun, which is a child of `/dmsDemo`.

Example A–2 BasicBinomial.java: Using Noun.create to Organize Sensors

```
private Noun binRoot;                // Container for Binomial series DMS metrics
Noun base = Noun.create("/dmsDemo");
binRoot   = Noun.create(base, "BasicBinomial", "MathSeries");
```

`/dmsDemo` is the noun root for the binomial sample application. The initial slash character marks it as the root.

The `BasicBinomial` noun is created with a noun type of `MathSeries` (the third parameter for the `create` method). The Noun type is a name that reflects the set of metrics being collected. In this case, `MathSeries` represents the metrics collected for the sample binomial application.

In the ImprovedBinomial servlet, located in the same binomial application, there is a similar set of lines (Example A–3). The only difference is that it creates a child noun called `ImprovedBinomial` instead of `BasicBinomial`.

Example A–3 ImprovedBinomial.java: Using Noun.create to Organize Sensors

```
private Noun binRoot;                // Container for Binomial series DMS metrics
Noun base = Noun.create("/dmsDemo");
binRoot   = Noun.create(base, "ImprovedBinomial", "MathSeries");
```

Note that the `dmsDemo` noun was not created with a noun type because its children are all nouns. It does not have any child sensors.

The Spy servlet displays all nouns of the same type as rows in a table; the name of the table is the noun-type. Metrics are represented as columns in the table.

It is good practice to only use Noun types for Nouns that directly contain Sensors. When a Noun contains only other Nouns, as in `dmsDemo`, and does not directly contain Sensors, the Spy servlet displays the Noun type as a metric table, with no metrics.

The dmsDemo Noun includes a Noun (BasicBinomial), but no direct Sensors. When the Noun type is not included for such a Noun, the Spy servlet does not display a metric table associated with the Noun.

> **Note:** Start Noun type names with a capital letter to distinguish them from other DMS names. See Section A.1.4, "DMS Naming Conventions" for details.

A.2.2.1 Choosing Noun Types

In general, you should give different names to nouns and noun types to provide a logical hierarchy for nouns of the same type at the same level. For example, in the dmsDemo application, there are two servlets: BasicBinomial and ImprovedBinomial. The DMS instrumentation uses the noun type MathSeries for both servlets. This noun type is created under /dmsDemo in the same hierarchy level for both servlets. Adhering to this practice makes the generated metric tables easier to understand. It also prevents some minimal information loss in the reporting process.

A.2.3 Defining and Using Metrics for Timing

To create metrics that measure the duration of a segment of code, you use PhaseEvent Sensors:

- Section A.2.3.1, "Defining PhaseEvent Sensors"
- Section A.2.3.2, "Using PhaseEvent Sensors"

A.2.3.1 Defining PhaseEvent Sensors

Example A–4 shows the DMS calls that declare and create the computeSeries PhaseEvent Sensor. This code defines a DMS metric named /dmsDemo/BasicBinomial/computeSeries.time.

Example A–4 Defining PhaseEvent Sensors

```
private PhaseEvent binComp; // Time to compute Binomial series
...
binComp = PhaseEvent.create(binRoot, "computeSeries", "Time to compute a Binomial series");
binComp.deriveMetric(Sensor.all);
```

PhaseEvent Sensors support a set of optional metrics, along with the default metric .time, which represents the time, as measured between the PhaseEvent start and the PhaseEvent stop calls. You can derive optional metrics with PhaseEvent Sensors individually or as a complete set. Table A–1 shows the available metrics for a PhaseEvent Sensor. The binComp.deriveMetric(Sensor.all) call in Example A–4 causes all the supported optional metrics to be computed and reported.

> **Note:** Using the method deriveMetric(Sensor.all) is recommended for adding optional metrics. Using this method with Sensor.all adds all metrics; this is good practice since the list of optional metrics could change in a future Oracle Fusion Middleware releases. In addition, the metrics are efficient to compute and are often useful in evaluating performance.

A.2.3.2 Using PhaseEvent Sensors

To use a PhaseEvent Sensor, an application calls the `start` method to indicate the beginning of a phase and subsequently calls the `stop` method to indicate the completion of the phase.

Example A–5 shows a code segment from `BasicBinomial.java` that uses the `start` and `stop` methods for the `/dmsDemo/BasicBinomial/computeSeries.time` metric. The `long` value named `token` that is returned from the PhaseEvent `start` method must be passed to the corresponding PhaseEvent `stop` method. This value is a timestamp representing the start time. Passing this value to the `stop` method enables DMS to compute the PhaseEvent duration.

> **Note:** To assure that PhaseEvents are stopped, each PhaseEvent `start` method, together with the code to be measured, should be in a `try` block with the PhaseEvent `stop` method in a corresponding `finally` block, as shown in Example A–5.

Example A–5 Using start and stop With PhaseEvent Sensors

```
long token = 0; // DMS
try {
    token = binComp.start(); // DMS
    BigInteger bins[] = bin(length);
    out.println("<H2>Binomial series for " +  length + "</H2>");
    for (int i = 0; i < length; i++)
        out.println("<br>" + bins[i]);
}
finally {
    binComp.stop(token); // DMS
    out.close();
}
```

Example A–5 shows code instrumented such that each time a phase starts, it is stopped (since the `stop` method is placed in the finally clause). This prevents runaway Phase Sensors; however, this can result in the time required to throw an exception possibly contributing to phase statistics. To prevent exception handling from impacting a PhaseEvent, use the `abort` method, as shown in Example A–6.

Example A–6 shows a code sample where a Phase that is not successfully stopped can be closed. The abort call removes the statistics corresponding to the corresponding start, and these statistics do not contribute to metric calculations.

Example A–6 Using abort with PhaseEvent Sensors

```
PhaseEvent pe = heavyPhase(param);
long token1 = 0;
long token2 = 0;
boolean stopped = false;
try {
      token1 = binComp.start();
      if (pe != null) token2 = pe.start();
      BigInteger bins[] = bin(length);
      out.println("<H2>ImprovedBinomial series for " +  length + "</H2>");
      for (int i = 0; i < length; i++)
        out.println("<br>" + bins[i]);
      if (pe != null) pe.stop(token2);
      binComp.stop(token1);
```

```
      stopped = true;
}
finally {
      if (!stopped) {
        if (pe != null) pe.abort(token2);
        binComp.abort(token1);
      }
}
```

A.2.4 Defining and Using Metrics for Counting

To create metrics that count the occurrences of an event, define and use an Event Sensor:

- Section A.2.4.1, "Defining Event Sensors"
- Section A.2.4.2, "Using Event Sensors"

A.2.4.1 Defining Event Sensors

Example A–7 shows the DMS calls that define an Event Sensor. This code allocates a counter and defines a DMS metric named `/dmsDemo/BasicBinomial/loops.count`.

Example A–7 Defining Event Sensors

```
private Event binLoop;   // Loops needed for Binomial series.
...
binLoop = Event.create(binRoot, "loops", "Iterations to compute series");
```

A.2.4.2 Using Event Sensors

DMS increments a counter when an application calls the `occurred` method for an Event Sensor. Example A–8 shows the `occurred` call for an Event Sensor that increments the `/dmsDemo/BasicBinomial/loops.count` metric.

Example A–8 Using occurred With Event Sensors

```
binLoop.occurred();
```

A.2.5 Defining and Using Metrics for Recording Status Information (State Sensors)

DMS captures status information with State Sensors. State Sensors track the value of Java primitives or the content of a Java Object. The supported types include integer, double, long, and object, as specified in the third argument to the `create` method. When a Java primitive State Sensor is updated with the wrong type, DMS attempts to convert the supplied value to the correct type. For object type State Sensors, DMS stores a reference to the object and by default calls `toString` on the object when the DMS value is sampled.

To create metrics that record status information, define and use State Sensors:

- Section A.2.5.1, "Defining State Sensors"
- Section A.2.5.2, "Using State Sensors"

A.2.5.1 Defining State Sensors

State Sensors support a default metric `value`, as well as optional metrics. You can define the `minValue` and `maxValue` optional metrics with State Sensors only if the State Sensor represents a numeric Java primitive (of type integer, double, or long). Table A–3 shows the available metrics for a State Sensor. Example A–4 shows how to enable optional metrics.

Example A–9 shows the DMS calls that declare and create a State Sensor. This code defines a DMS metric named `/dmsDemo/BasicBinomial/lastComputed.value`.

Example A–9 Defining State Sensors

```
private State binLast;   // Value of the last computed element in series.
...
binLast = State.create(binRoot, "lastComputed", State.OBJECT, "",
                           "Value of last computed series element");
```

When you define a State Sensor, use an empty string in the fourth argument to the `create` method if no value is associated with the State Sensor, otherwise use a string listing the appropriate values (see Example A–9). State Sensors are created without an initial value. If you need to check whether a State Sensor has been initialized, use the `isInitialized` method.

If you want your State Sensor to store the string value of an object, and not store a reference to the object, use the `setCopy` method with the value `TRUE`. This tells the State Sensor to store the result of calling `toString` on an object rather than using a reference to the object for the metric value.

A.2.5.2 Using State Sensors

When an application calls a State Sensor's `update` method, DMS updates the value of the State Sensor. Example A–10 shows the `update` call for a State Sensor that updates the `/dmsDemo/BasicBinomial/lastComputed.value` metric.

Example A–10 Using update With State Sensors

```
binLast.update(bins[k-1].toString());
```

A.3 Validating and Testing Applications Using DMS Metrics

You should test and verify the accuracy of the metrics that you add to Java applications. This section includes the following subsections:

- Section A.3.1, "Validating DMS Metrics"
- Section A.3.2, "Testing DMS Metrics For Efficiency"

A.3.1 Validating DMS Metrics

Use `wlst` and other available DMS monitoring tools to verify and test new metrics.

Try to validate the following for new metrics:

- Do expected metrics appear in the display? Test this by examining the code to ensure that all the metric names added using DMS instrumentation appear in your display or saved set of metrics.
- Do unexpected metrics appear in the display? Verify that you have only added the metrics that you planned to add.

- Are the metric values you see within reasonable ranges? Usually, upper and lower bounds for metrics can be established. You then test that the reported values for metrics do not exceed the expected bounds.

  For example, a "size of pool" metric should never report a negative value.

- Ensure the new metrics are needed. For example, if you add a PhaseEvent that always measures an event of very short duration, consider changing the metric to an Event metric, or remove the metric.

- Ensure the new metrics are accurate. For most applications using DMS metrics, accuracy is more important than the performance cost of adding the DMS instrumentation. New DMS metrics should provide reliable and useful information.

  Testing for accuracy can be difficult. However, if an alternate means of measuring a particular metric is available then use it to verify metric values. For example, if you submit a known number of requests to a server and measure total time for the experiment, then you predict correct values for the relevant metrics and compare them with the actual monitored values. As another example, you can verify an Event Sensor `count` metric by examining records that you write to a log file or to the console.

  Check for timing inaccuracies that may apply for the metrics. Timing inaccuracies may be caused when low-resolution clocks time metrics for an interval of short duration. For example on Windows systems, the default Java clock advances only once every 15 milliseconds. DMS metrics reported for brief events on these systems must be analyzed with care. Consider using the high resolution clock to address this issue.

  See Also: Section A.9, "Using a High Resolution Clock to Increase DMS Precision"

A.3.2 Testing DMS Metrics For Efficiency

The use of DMS metrics has some influence on application performance. When adding metrics, note the following:

- The processing required for computing and storing metrics can slow down the execution of an application. DMS is fast, but it may have some required performance cost. In addition, DMS cannot prevent developers from using the DMS API inefficiently. Therefore, before adding DMS instrumentation, establish reasonable expectations. After completing the implementation, measure the actual costs and compare them to your expectations. Be prepared to make changes to the instrumentation to reduce performance impacts until the measurements agree with expectations.

- DMS provides the `DMSConsole.getSensorWeight` method to help you control the use of metrics. The central setting is an advisory measurement level that DMS does not enforce. To control which metrics to include, at run time, the code must test the value for `SensorWeight` to determine whether to make DMS calls. See Section A.5, "Conditional Instrumentation Using DMS Sensor Weight" for details.

- When integrating DMS instrumentation with an existing package or when implementing a new feature, you should consider insulating a previously working system. For example, you could include an option to enable and disable new DMS metrics.

- You should run your performance tests with and without DMS enabled. If your tests show unacceptable results with DMS enabled, then you may want to re-design or re-implement metrics.

A.4 Understanding DMS Security Considerations

DMS metrics do not support user-based access to DMS reports. When you define and use a DMS metric, the metric is available to any administrator that has access to DMS metrics. This means when you add DMS metrics, it is good practice to avoid placing customer-sensitive information in the metrics.

A.5 Conditional Instrumentation Using DMS Sensor Weight

Use the DMS Sensor weight feature to conditionally limit your instrumentation. With Sensor weight, you specify that applications execute expensive instrumentation only when the Sensor weight is set to a particular value. Using this feature enables you to include expensive metrics that you may only need for debugging.

Example A–11 shows how to use `DMSConsole.getSensorWeight` and optionally to define and use a metric. SensorWeight should not be used to control the content of a noun (sensors or metrics); sensor-weight should be used (as in the example below) to control the existence of a noun, and more importantly to control the existence of nouns of a particular type.

The Sensor weight is set globally using the `oracle.dms.sensors` property on the command-line. Set this property using the startup options. Supported values for this property include: `none`, `normal`, `heavy`, and `all`.

Example A–11 Using SensorWeight for Conditional Instrumentation

```
/* DMS Method
 *
 * If the SensorWeight is high enough, return a phase with the
 * parameter in the name. Otherwise, return null.
 */
PhaseEvent heavyPhase(String param) {
   PhaseEvent pe = null;
   if (DMSConsole.getSensorWeight() > DMSConsole.NORMAL) {
      Noun base = Noun.create(binRoot, param, "MathSeries");
      pe = PhaseEvent.create(base, "computeSeries",
                                   "Time to compute a Binomial series");
      pe.deriveMetric(Sensor.all);
   }
   return pe;
}
```

A.6 Dumping DMS Metrics to Files

In a Java application, use the following method to dump DMS metrics to a file.

The following code enables you to append or replace the contents of the specified file with the current metrics:

```
DMSConsole cons2 = new DMSConsole();
DMSConsole.dump("dmsmathseries.log", "xml", true);
```

The first argument specifies the file path name, the second argument specifies the output format, and the third argument specifies if the output is appended to the file or replaces the contents of the file.

WARNING: This process dumps all DMS information for the JVM. In cases where the JVM is running several applications (such as J2EE) this process dumps all metrics for all applications.

A.7 Resetting and Destroying Sensors

The `Sensor` abstract class provides methods to control PhaseEvent, Event, and State Sensors. The `reset` method resets a Sensor's metrics to initial values. The `getResetTime` method determines if a Sensor has been reset. The `destroy` method removes a Sensor from DMS and releases references to its underlying resources.

Note: Do not use these methods to reset or destroy built-in metrics. The `reset` and `destroy` methods are intended for use with metrics that you create. Fusion Middleware Control and other Oracle Fusion Middleware administrative facilities could report unexpected values or have unexpected behavior if you use these methods on internal built-in metrics.

A.8 DMS Coding Recommendations

The following list includes coding recommendations for working with DMS.

1. There is a global name space for DMS metrics. When you create a new Noun Sensor (PhaseEvent, Event, or State), its full name must not conflict with names in use by Oracle built-in metrics, or by other applications. It is therefore a good idea to have a root Noun for your application that contains the application's full name. This prevents name space collisions.

   See Also: Section A.1.4.1, "General DMS Naming"

2. Be sure all PhaseEvents are stopped. If the code block to be measured is not in a `try` block, then put it in a `try` block that includes PhaseEvent's `start`. Put the PhaseEvent's `stop` in a `finally` block. Alternatively, make use of the `abort` method in the `finally` block, as shown in Example A–6.

   See Also: Section A.2.3.2, "Using PhaseEvent Sensors"

3. Follow DMS naming conventions.

   See Also: Section A.1.4, "DMS Naming Conventions"

4. Avoid creating any DMS Sensor or Noun more than once. The DMS API enables this, and avoids creation of multiple objects, but DMS performs lookups for each subsequent creation attempt. Thus, whenever possible, you should define Sensors and Nouns during static initialization, or in the case of a Servlet, in the `init` method.

5. Assign a type for each Noun that contains Sensors. If no type is assigned, the type is given the value "n/a" (not available). Nouns with the type specified as "n/a" are not shown in the Spy servlet.

6. Only use PhaseEvents to measure a section of code that is expensive to execute, and takes a significant time to execute under some conditions. In the case where the code never takes significant time to execute, use an Event metric, or remove the PhaseEvent.
7. The DMS API calls are threadsafe; they provide sufficient synchronization to prevent races and access bugs.
8. J2EE applications should not control the lifecycle of the DMS Console - the lifecycle of the DMS Console is managed by the J2EE container. DMS is global to the J2EE server's JVM and therefore changes made to DMS can affect all applications running in that J2EE container.

Note: The beginning part of the execution context string generated by Java API is in IP address format which is different that the string generated by C API. To avoid issues, ensure that the same ECID is used throughout a single request/transaction. The ECID can be in any format as long as it is used consistently.

A.8.1 Isolating Expensive Intervals Using PhaseEvent Metrics

Carefully consider the requirements for new metrics when you add DMS instrumentation. It is important to add a sufficient number of metrics to validate that your code is working as expected.

Try to observe the following guidelines when you add DMS metrics:

1. Add PhaseEvent Sensors only to provide an overview of the time the system spends in your block of code or module. You do not need to collect performance data for every method call, or for every distinct phase of your code or module.
2. When your code calls external code that you do not control, and that you expect could take a significant amount of time, add a PhaseEvent Sensor to track the start and the completion of the external code.

Following these guidelines for adding PhaseEvent metrics provides the following benefits:

- Helps limit the amount of information that DMS collects.
- Enables those analyzing the system to prove that a module gives the expected run time performance.
- Ensures that people viewing DMS metrics can validate run time performance without seeing an overwhelming amount of data.
- Enables those analyzing system performance to separate and track your module from other system modules that are either expensive or failure prone.

A.9 Using a High Resolution Clock to Increase DMS Precision

By default DMS uses the system clock for measuring time intervals during a `PhaseEvent`. The default clock reports microsecond precision in C processes such as Apache and reports millisecond precision in Java processes. Optionally, DMS supports a high resolution clock to increase the precision of performance measurements and lets you select the values for reporting time intervals. You can use a high resolution clock when you need to time phase events more accurately than is possible using the default clock or when the system's default clock does not provide the resolution needed for your requirements.

Note: The resolution of the clocks (both default and high resolution) is platform-dependent. On some systems the default clock may not provide sufficient resolution for timing requirements. In particular, on Windows platforms, many users request greater precision than the default clock provides, because it advances only once every 15 milliseconds. DMS metrics reported for brief events on these systems must be analyzed with care. Consider using the high resolution clock to address this issue.

Also note that the accuracy of high resolution clocks are also platform-dependent. Selecting a high resolution clock does not guarantee that the system reports nanosecond times accurately.

This section covers the following topics:

- Section A.9.1, "Configuring DMS Clocks for Reporting Time for Java"
- Section A.9.2, "Configuring DMS Clocks for Reporting Time for Oracle HTTP Server"

A.9.1 Configuring DMS Clocks for Reporting Time for Java

Selecting the high resolution clock changes clocks for all applications running on the process where the clock is changed. You set the DMS clock and the reporting values globally using the `oracle.dms.clock` and `oracle.dms.clock.units` properties, which control process startup options.

For example, to use the high resolution clock with the default values, set the following property on the Java command line:

```
-Doracle.dms.clock=highres
```

Caution: If you use the high resolution clock, the default values are different from the value that Fusion Middleware Control expects (msecs). If you need the Fusion Middleware Control displays to be correct when using the high resolution clock, then you need to set the units property as follows:

```
-Doracle.dms.clock.units=msecs
```

Table A–6 shows supported values for the `oracle.dms.clock` property.

Table A–7 shows supported values for the `oracle.dms.clock.units` property.

Table A–6 oracle.dms.clock Property Values

Value	Description
DEFAULT	Specifies that DMS use the default clock. With the default clock, DMS uses the Java call `java.lang.System.currentTimeMillis` to obtain times for PhaseEvents. The default value for the units for the default clock is MSECS.
HIGHRES	The Java Highres clock uses `System.nanoTime()` (no JNI required).

Table A–7 oracle.dms.clock.units Property Values

Value	Description
MSECS	Specifies that the time be converted to milliseconds and reported as "msecs". A millisecond is 10^{-3} seconds. **Note**: This is the default value for the default clock.
USECS	Specifies that the time be converted to microseconds and reported as "usecs". A microsecond is 10^{-6} seconds.
NSECS	Specifies that the time be converted to nanoseconds and reported as "nsecs". A nanosecond is 10^{-9} seconds. **Note**: This is the default value for the high resolution clock.

Note the following when using the high resolution DMS clock:

- When you set the `oracle.dms.clock` and the `oracle.dms.clock.units` properties, any combination of upper and lower case characters is valid for the value that you select (case is not significant). For example, any of the following values are valid to select the high resolution clock: highres, HIGHRES, HighRes.
- DMS checks the property values at startup. When you set the clock with a value that does not match those listed in Table A–6, then DMS uses the default clock. If the `oracle.dms.clock` property is not set, DMS also uses the default clock.
- If the specified clock units property value does not match those listed in Table A–7, then DMS uses the default units for the specified clock. If the `oracle.dms.clock.units` property is not set, DMS uses the default units for the specified clock.

A.9.2 Configuring DMS Clocks for Reporting Time for Oracle HTTP Server

The default clock for measuring Oracle HTTP Server performance has a resolution of microseconds (usecs). You can optionally select a higher resolution clock to monitor C processes running under Oracle HTTP Server. To use the high resolution clock under Oracle HTTP Server, you need to set configuration options in httpd.conf, or specify environment variables on the command line.

Table A–8 lists the environment variables that control the Oracle HTTP Server DMS clock. Table A–9 describes the httpd.conf configuration options that control the Oracle HTTP Server DMS clock. If you set both the command line options and the httpd.conf configuration options, the configuration options override the values set on the command line.

Table A–8 Oracle HTTP Server DMS Clock Environment Variables

Environment Variable	Description
DMS_CLOCK	Specifies the clock to use for DMS timing. The values are interpreted the same as with oracle.dms.clock. Valid Values: DEFAULT, HIGHRES
DMS_CLOCK_UNITS	Specifies the units for reporting DMS timing values. The values are Interpreted the same as with oracle.dms.clock.units. Valid Values: MSECS, NSECS, USECS Default Value: USECS

Table A–9 Oracle HTTP Server DMS Clock Configuration Parameters

Parameter	Description
`DmsClock`	Specifies the clock for HTTP listener processes started by Oracle HTTP Server, as the oracle.dms.clock property does for Java processes. Valid Values: DEFAULT, HIGHRES
`DmsClockUnits`	Specifies the time units for HTTP listener processes started by Oracle HTTP Server, as the oracle.dms.clock.units property is for Java processes. Valid Values: MSECS, NSECS, USECS Default Value: USECS

With these options DMS uses a high resolution clock for all the Oracle HTTP Server processes that it monitors, and DMS reports values using the milliseconds units (msecs).

Caution: If you are using the high resolution clock for the Oracle HTTP Server, the default units are NSECS on most platforms. If you need to use Fusion Middleware Control, it expects USECS for the units. If you need the Fusion Middleware Control displays to be correct when using the high resolution clock, then you need to set the units property as follows:

```
DmsClock=HIGHRES
DmsClockUnits=USECS
```

A.10 Rolling Up DMS Data for Descendent Nouns

Oracle Fusion Middleware includes the DMS Roll-up feature which enables you to specify metric aggregation. You can use the Roll-up feature to specify metric aggregation during DMS instrumentation; roll-up is specified to apply to descendents of a specified noun type. You can specify whether the roll-up should only apply to direct descendents or to all descendents. Example A–12 shows code that generates a DMS tree, as represented in Figure A–2. Each noun of type `myContainer` contains the `percentageFull`, `close`, and `open` Sensors (see Figure A–2).

Note: The code in Example A–12 generates a noun tree hierarchy that violates the guidance described in Section A.2.2.1, "Choosing Noun Types". In this example, it makes sense for some nouns to have descendents and ancestors of the same noun type. The roll-up feature described in this section can collect data which might otherwise be lost.

Example A–12 DMS sample code creating noun hierarchy of metrics

```
// Create DMS Noun hierarchy for metrics
Noun home = Noun.create(Noun.getRoot(), "Home", "myContainer");
Noun containers = Noun.create(home, "Containers", "myContainer");
Noun closets = Noun.create(containers, "Closets", "myContainer");
Noun bedrooms = Noun.create(closets, "Bedrooms", "myContainer");
Noun br1 = Noun.create(bedrooms, "BR1", "myContainer");

// Create a closet Noun and create Sensors for it
```

```
Noun c1 = Noun.create(br1, "C1", "myContainer");
State percent = State.create(br1, "percentageFull", State.INTEGER, "percent",
"percentage full");
Event close = Event.create(br1, "close", "container closed");
PhaseEvent open = PhaseEvent.create(br1, "open", "open container");

// Derive metrics for State and PhaseEvent Sensors
percent.deriveMetric(Sensor.all);
open.deriveMetric(Sensor.all);
```

Figure A–2 Containers DMS Hierarchy Showing Tree Containing Metrics

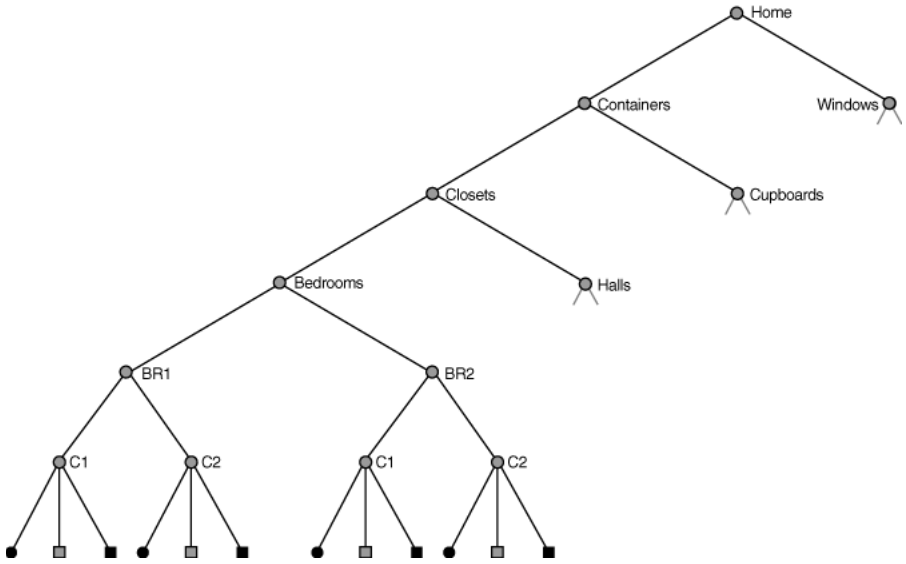
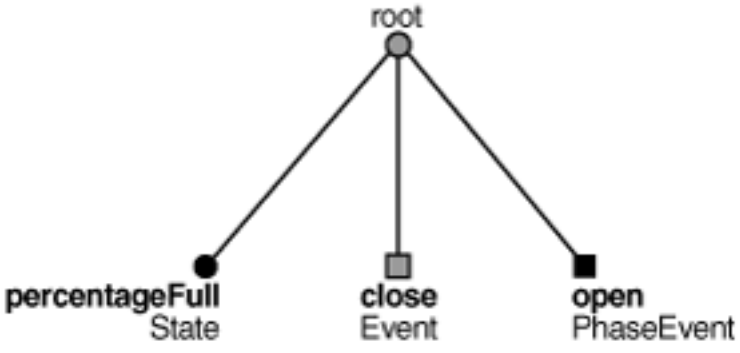


Figure A–3 shows a tree with a set of descendent containers. The nouns `C1` and `C2` under the bedrooms `BR1` and `BR2` are of type `myContainer` (see Figure A–3 for a description of `myContainer` metrics).

Figure A–3 Noun myContainer showing Sample Sensors



The roll-up feature enables you to aggregate a summary for descendent Nouns. For example, you can add the roll-up call to a `bedrooms` noun, as shown in Example A–12. To aggregate `myContainer` type metrics under `BR1`, use the following call:

```
br1.rollup("myContainer", Noun.DIRECT);
```

This call creates a roll-up noun named `myContainer_rollup` under `/Home/Containers/Closets/Bedrooms/BR1`. The roll-up noun contain the same sensors as the associated noun, including: `percentageFull`, `close`, and `open`.

DMS roll-up metrics let you roll-up the sensors in all descendent nouns of the given types or only those in the direct descendent nouns. Specifying `Noun.DIRECT` in the roll-up call aggregates only direct descendent nouns of the specified type. To aggregate the metrics from all descendent nouns of type `myContainer` instead, use a call such as the following including `Noun.ALL`:

```
closets.rollup("myContainer", Noun.ALL);
```

Roll-up metrics include aggregate summary information for their contents. Table A–10 shows the available derived rollup metrics for each Sensor type.

Table A–10 Roll-up Metrics Included Derived Metrics

Metric	Description
PhaseEvent	The derived metrics for a PhaseEvent rollup metric include the following: ■ `time`: the sum of `time` metrics. ■ `completed`: the sum of the `completed` metrics. ■ `maxTime`: the maximum of the `maxTime` metrics. ■ `minTime`: the minimum of the `minTime` metrics: ■ `avg`: the average time computed for all Sensors. ■ `active`: the sum of the `active` metrics.
Event	The derived metrics for a Event roll-up metrics include the following: ■ `sum`: the total of all `count` metrics. ■ `avg`: the average of all `count` metrics.
State	The derived metrics for a State rollup metrics include the following: ■ `sum`: the total of all `value` metrics. ■ `avg`: the average of all value metrics. ■ `maxValue`: the maximum of the `maxValue` metrics. ■ `minValue`: the minimum of the `minValue` metrics.
`descendents`	The roll-up noun includes a `descendents` state sensor that reports whether the roll-up covers only direct descendents or all descendents.
`rolled`	The roll-up noun includes a `rolled` state sensor, which reports the number of nouns that are rolled up.
`refresh`	The roll-up noun includes a `refresh` phase event, which reports the time spent aggregating the metrics for this rollup noun.

Example A–13 shows sample metrics created for the `myContainer` roll-up noun under `/Home/Containers/Closets`.

Example A–13 Test

```
myContainer_rollup
  descendent.value: all
  percentageFull.sum 40 percent
  percentageFull.avg 10.0 percent
  percentageFull.min 1 percent
  percentageFull.max 29 percent
  close.sum: 3
```

```
close.avg: 0.75
open.time: 871 msecs
open.completed: 4 ops
open.maxTime: 722 msecs
open.minTime: 23 msecs
open.avg: 217.7 msecs
open.active: 0
rolled.value: 4 nouns
refresh.maxActive:   1 threads
refresh.active:      0 threads
refresh.avg: 0.2857142857142857 msecs
refresh.maxTime:     1 msecs
refresh.minTime:     0 msecs
refresh.completed:   7 ops
refresh.time:        2 msecs
```

Note that the metrics are similar to the `myContainer` metrics. The roll-up metrics have several key differences, as follows:

1. The roll-up noun contains the `descendent`, `rolled`, and `refresh` metrics (see Table A–10 for details).
2. The `percentageFull` State contains `sum` and `avg` metrics rather than the `value` metric. The name of each metric reflects its content.
3. The `close Event` contains `sum` and `avg` metrics rather than the `count` metric. The name of each metric reflects its content.
4. The `open PhaseEvent` does not contain a `maxActive` metric as it would have no meaning in this context.

See Also: *Oracle Fusion Middleware DMS Java API Reference*

B

Related Reading and References

All of the external documentation and web site references made in this book are listed in this appendix.

- Section B.1, "Oracle Documentation"
- Section B.1.1, "Oracle Fusion Middleware Library"
- Section B.1.2, "Oracle Database"
- Section B.1.3, "Oracle JRockit Java Virtual Machine (JVM)"
- Section B.2, "Sun Microsystems Information"

B.1 Oracle Documentation

`http://www.oracle.com/technology/documentation/index.html`

- Section B.1.1, "Oracle Fusion Middleware Library"
- Section B.1.2, "Oracle Database"
- Section B.1.3, "Oracle JRockit Java Virtual Machine (JVM)"

B.1.1 Oracle Fusion Middleware Library

`http://www.oracle.com/technology/documentation/middleware.html`

- Section B.1.1.1, "Cross-Suite Administration Guides"
- Section B.1.1.2, "WebCenter"
- Section B.1.1.3, "Identity Management"
- Section B.1.1.4, "SOA Suite"

B.1.1.1 Cross-Suite Administration Guides

Oracle Fusion Middleware Security Guide

Oracle Fusion Middleware Concepts

Oracle Fusion Middleware Administrator's Guide

Oracle Fusion Middleware High Availability Guide

Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management

Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite

Oracle Fusion Middleware Enterprise Deployment Guide for Oracle WebCenter

Oracle Fusion Middleware Security Overview

B.1.1.2 WebCenter

Oracle Fusion Middleware Getting Started with Oracle WebCenter

Oracle Fusion Middleware Developer's Guide for Oracle WebCenter

Oracle Fusion Middleware User's Guide for Oracle WebCenter

Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers

Oracle Fusion Middleware Tutorial for Oracle WebCenter Spaces Users

Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter

B.1.1.3 Identity Management

Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory

Oracle Fusion Middleware Integration Guide for Oracle Identity Management

Oracle Fusion Middleware User Reference for Oracle Identity Management

Oracle Fusion Middleware Getting Started with Oracle Identity Management

Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory

Oracle Fusion Middleware Application Developer's Guide for Oracle Identity Management

Oracle Fusion Middleware Tutorial for Oracle Identity Management

Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation

B.1.1.4 SOA Suite

Oracle Fusion Middleware Getting Started with Oracle SOA Suite

Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite

Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite

Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring

Oracle Fusion Middleware User's Guide for Technology Adapters

Oracle Fusion Middleware Tutorial for Running and Building an Application with Oracle SOA Suite

B.1.2 Oracle Database

http://www.oracle.com/technology/documentation/database.html

Oracle Database Performance Tuning Guide

Oracle Database Administrator's Guide

Oracle Database 2 Day DBA

Oracle Database Concepts

B.1.3 Oracle JRockit Java Virtual Machine (JVM)

"Welcome to Oracle JRockit" at
http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html

"First Steps for Tuning the Oracle JRockit JVM " at
http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/bestpractices.html

"Tuning the Memory Management System" at
http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/memman.html#wp1087125

"About Profiling and Performance Tuning" at
http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/about_prof_perftune.html

B.2 Sun Microsystems Information

- For general information about Sun Microsystems, see Sun's Web site at http://www.sun.com
- Sun Microsystems Performance Information at http://java.sun.com/docs/performance/index.html
- Java Standard Edition Platform Documentation at http://java.sun.com/docs/index.html
- Java 2 SDK, Standard Edition Documentation http://java.sun.com/javase/6/docs
- "Solaris Tunable Parameters Reference Manual" at http://docs.sun.com/app/docs/doc/819-2724?
- For more about Solaris configuration, check the Solaris FAQ at http://java.sun.com/products/jsp/faq.html.

B.2.1 Sun Java HotSpot Virtual Machine

"Java HotSpot Garbage Collection" at
http://java.sun.com/javase/technologies/hotspot/gc/index.jsp

"Memory Management in the Java HotSpot Virtual Machine" at
http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf

"Diagnosing a Garbage Collection Problem" at
http://java.sun.com/docs/hotspot/gc1.4.2/example.html

Index

A

Advanced Replication-based replication
  tuning, 17-3
ANALYZE function of DBMS_STATS
    package, 17-16
attributes
  garbage collection
    tuning, 17-22
  orclskewedattribute, 17-13
  skewed, optimizing searches for, 17-13
  tuning, 17-3, 17-4
automatic restart, 23-3

B

bulk tools
  performance, 17-2
bulkdelete command
  performance tuning, 17-2
  tuning database parameters, 17-11
bulkload command
  on RAC database, 17-7
  performance tuning, 17-3
  tuning, 17-11

C

cache
  server entry, 17-7
caches
  entry
    tuning, 17-20
  privilege group membership
    tuning, 17-20
change logs
  enable or disable generation
    tuning, 17-20
  generation, tuning, 17-3
  purging
    tuning, 17-6
  purging configuration entry
    tuning, 17-22
  when to disable generation, 17-11
conflict resolution
  tuning, 17-6
connection timeout
  tuning, 17-10, 17-20
connections to database
  tuning, 17-20

D

database
  connections
    tuning, 17-20
  queries, optimization of, 17-13
  redo log files
    tuning, 17-11
  SQL execution plans
    performance impact, 17-5
  statistics
    tuning, 17-4
Database Statistics Collection Tool
  tuning, 17-18
db_cache_size database instance parameter
  tuning, 17-3
DBMS_STATS package, 17-16
death detection, 23-3
dedicated LDAP server for Oracle Directory
    Integration Platform and replication
  tuning, 17-7
directives
  See also httpd.conf directives
Disaster Recovery, 23-3
dispatcher
  maximum server response time, 17-20
dispatcher threads
  number
    tuning, 17-20
DMS
  coding tips, A-16
  conditional instrumentation, A-15
  Event sensors, A-5
    using, A-11
  getSensorWeight, A-15
  instrumentation
    definition of, A-2
    using, A-8
  metrics
    definition of, A-4
    dumping to files, A-16

monitoring metrics, A-2
naming conventions, A-6
nouns, A-3, A-5
naming conventions, A-7
PhaseEvent sensors, A-4
using, A-10
rollup
descendents, A-23
refresh, A-23
rolled, A-23
rollup nouns, A-6
security, A-15
sensors, A-3
definition of, A-4
destroying, A-16
resetting, A-16
State sensors, A-5
using, A-12
terminology, A-3
testing metrics, A-14
validating metrics, A-13
DNS
domain name server, 5-7

E

entries
returned by a search
tuning, 17-10
entry cache, 17-7
optimizing, 17-8
tuning, 17-20
when to use, 17-7
Event sensors, A-5
event tracking
tuning, 17-9

F

Failover, 23-3
failover, 23-3
on RAC database, 17-7
Fusion Middleware Control
performance attributes
instance-specific, 17-20
shared, 17-20

G

garbage collection
tuning, 17-6
group cache, enabling
tuning, 17-10
group entries
large
optimizing searches for, 17-12

H

high LDAP write operations load
tuning, 17-11
HIQ
See human intervention queue
HostNameLookups
directive, 5-7
httpd.conf
directives
HostNameLookups, 5-7
KeepAlive, 5-5
KeepAliveTimeout, 5-5
ListenBacklog, 5-3
MaxClients, 5-3
MaxKeepAliveRequests, 5-5
MaxRequestsPerChild, 5-4
Timeout, 5-5
human intervention queue
changes
tuning, 17-6

I

idle connection timeout
tuning, 17-3, 17-20
in-memory processing of search filters, 17-13
iostat utility performance evaluation, 17-16

J

job_queue_processes database instance parameter
tuning, 17-3

K

KeepAlive directive, 5-5
KeepAlive httpd.conf directive, 5-5
KeepAliveTimeout httpd.conf directive, 5-5

L

large group entries
optimizing searches, 17-12
ldapmodify command
modifying attributes
tuning, 17-21
ListenBacklog httpd.conf directive, 5-3
load balancing
on RAC database, 17-7
logging
performance implications of, 5-6

M

matchDN information
tuning, 17-4
max_commit_propagation_delay database instance parameter
tuning, 17-3
MaxClients directive, 5-3
maximum entry cache size
tuning, 17-20
MaxKeepAliveRequests httpd.conf directive, 5-5
MaxRequestsPerChild httpd.conf directive, 5-4

mpstat utility performance evaluation, 17-16

N

naming conventions
DMS, A-6
network retry timeout
tuning, 17-20
nouns
DMS, A-5
naming conventions, A-7
rollup, A-6
type, A-5
number of processes and threads
tuning, 17-3

O

ODSM
configuring changelog purging, 17-23
navigating to the changelog purging configuration entry, 17-23
oidstats.sql
Database Statistics Collection Tool, 17-18
optimizing
See also tuning
optimizing searches, 17-12
large group entries, 17-12
Oracle Directory Integration Platform
tuning, 17-5
Oracle Process Manager and Notification Server (OPMN), 4-9
Oracle WebLogic Scripting Tool (WLST)
See Also WLST commands
orclchangeretrycount attribute
tuning, 17-6
orclconflresolution attribute
tuning, 17-6
orcldispthreads attribute, 17-20
orclecacheenabled attribute, 17-20
server entry cache optimization, 17-9
orclecachemaxentries attribute, 17-20
server entry cache optimization, 17-9
orclecachemaxentsize attribute, 17-20
optimizing searches for large group entries, 17-12
server entry cache optimization, 17-9
orclecachemaxsize attribute, 17-20
server entry cache optimization, 17-9
orclenablegroupcache attribute
tuning group cache, 17-10
orclgeneratechangelog attribute, 17-20
tuning, 17-4
orclhiqschedule attribute
tuning, 17-6
orclinmemfiltprocess attribute
examples, 17-14
optimizing performance of complex search filters, 17-13
orclldapconntimeout attribute, 17-20
tuning, 17-3, 17-4
orclmatchdnenabled attribute
tuning, 17-4
orclmaxcc attribute, 17-20
server entry cache optimization, 17-9
tuning, 17-3, 17-4
orclmaxconnincache attribute, 17-20
orclmaxldapconns attribute, 17-20
orclmaxserverresptime attribute, 17-20
orclnwrwtimeout attribute, 17-20
tuning connection timeout, 17-11
orcloptrackmaxtotalsize attribute
security event tracking, 17-9
orcloptracknumelemcontainers attribute
security event tracking, 17-9
orclpluginworkers attribute, 17-20
orclpurgeinterval attribute
tuning, 17-7, 17-22
orclpurgetargetage attribute
tuning, 17-7, 17-22
orclserverprocs attribute, 17-20
server entry cache optimization, 17-9
tuning, 17-3, 17-4
orclskewedattribute attribute, 17-13
optimizing searches for skewed attributes, 17-13
orclskiprefinsql attribute
tuning, 17-3, 17-4
orclthreadspersupplier attribute
tuning, 17-6
orclupdateschedule attribute
tuning, 17-6

P

parameters
KeepAlive, 5-5
KeepAliveTimeout, 5-5
ListenBacklog, 5-3
MaxClients, 5-3
MaxKeepAliveRequests, 5-5
MaxRequestsPerChild, 5-4
Timeout, 5-5
password policies
tuning, 17-7
performance
attributes
modifying by using Fusion Middleware Control, 17-20
See also tuning
tuning, tools for, 17-16
persistent connections
KeepAlive directive, 5-5
pga_aggregate_target database parameter
tuning bulkload, 17-3, 17-11
PhaseEvent sensors, A-4
plug-in threads
per server process
tuning, 17-20
process death detection, 23-3
processes database instance parameter
tuning, 17-3

Q

queries, database
  optimizing, 17-13

R

RAC
  failover, 17-7
  load balancing, 17-7
RAC database
  bulkoad command, 17-7
  deployments
    tuning, 17-3
  tuning Oracle Internet Directory with, 17-7
redo log files
  tuning, 17-11
replication
  performance tuning, 17-5
  tuning, 17-5
retry timeout
  tuning, 17-20
rollup
  DMS, A-20

S

sar utility performance evaluation, 17-16
search
  filters
    processing in memory, 17-13
  number of entries returned
    tuning, 17-10
security event tracking
  tuning, 17-9
Server, 23-3
server attributes
  tuning, 17-3
server entry cache, 17-7
  attribute settings, 17-8
  optimizing, 17-8
  tuning, 17-7
  when to use, 17-7
server load balancing, 23-3
server processes
  number
    tuning, 17-20
server response time
  maximum
    tuning, 17-20
server response to dispatcher
  tuning, 17-20
session_cached_cursors database instance parameter
  tuning, 17-3
SGA Auto Tuning, 17-2
sga_max_size database instance parameter
  tuning, 17-2
sga_target database parameter
  tuning, 17-2
  tuning bulkdelete, 17-11
shared_pool_size database instance parameter
  tuning, 17-3
sizing wizard, 17-16
skewed attributes, 17-13
  optimizing searches for, 17-13
skip referral for search
  tuning, 17-3
state replication and routing, 23-3
State sensors, A-5

T

threads
  plug-in
    tuning, 17-20
timeout
  connection
    tuning, 17-20
Timeout httpd.conf directive, 5-5
tools
  evaluating performance, 17-16
  for tuning, 17-16
top utility
  performance evaluation, 17-16
tuning
  advanced configurations, 17-4
  Advanced Replication-based replication, 17-3
  bulkload, 17-11
  change log generation, 17-11
  database connections, 17-20
  dispatcher threads, 17-20
  enable or disable change log generation, 17-20
  enable or disable entry cache, 17-20
  high LDAP write operations load, 17-11
  idle connection timeout, 17-20
  maximum entries in entry cache, 17-20
  maximum plug-in threads per server
    process, 17-20
  modifying attributes
    by using Fusion Middleware Control, 17-20
    by using ldapmodify command, 17-21
  modifying garbage collection attributes, 17-22
  network retry timeout, 17-20
  number of processes and threads, 17-3
  oidstats.sql, 17-18
  optimizing searches for large group entries, 17-12
  optimizing searches for skewed attributes, 17-13
  password policies, 17-7
  performance evaluation tools, 17-16
  privilege group membership cache
    tuning, 17-20
  RAC Database deployments, 17-3
  recommendations
    tuning and sizing wizard, 17-16
  replication, 17-5
  security event tracking, 17-9
  server attributes, 17-3
  server entry cache, 17-7
  server processes, 17-20
  server response time
    maximim, 17-20

SGA Auto Tuning, 17-2
skewed attributes
optimizing searches, 17-13
timeout for write operations, 17-10
tools, 17-16
verifier policies, 17-7
when replication or Directory Integration Platform deployed, 17-5
tuning and sizing wizard, 17-16
tuning Database Statistics Collection Tool, 17-18
tuning entries to be returned by a search, 17-10

U

undo tablespace size
tuning for high LDAP write operations load, 17-11
UTLBSTAT.SQL, 17-16
UTLESTAT.SQL, 17-16

V

verifier policies
tuning, 17-7
vmstat utility performance evaluation, 17-16

W

Windows
Task Manager
evaluating performance, 17-16
Windows Performance Monitor
performance evaluation, 17-16
wizards
tuning and sizing, 17-16
write operation timeout
tuning, 17-10