

Oracle® Fusion Middleware

Tuning Performance Guide

12c (12.1.3)

E42117-03

January 2015

Describes how to monitor, optimize performance, and configure components for optimal performance in the Oracle Fusion Middleware environment.

Oracle Fusion Middleware Tuning Performance Guide 12c (12.1.3)

E42117-03

Copyright © 2013, 2015 Oracle and/or its affiliates. All rights reserved.

Primary Author: Hannah Cheng

Contributing Author: Peter Laquerre, Lisa Jamen, Jayaram Kasi

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	xi
Audience	xi
Documentation Accessibility	xi
Conventions	xi
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 Identifying Top Performance Areas	2-1
2.2 Securing Sufficient Hardware Resources	2-2
2.3 Tuning the Operating System	2-3
2.4 Tuning Java Virtual Machines (JVMs)	2-3
2.5 Tuning the WebLogic Server	2-4
2.6 Tuning Database Parameters	2-4
2.6.1 Tuning Database Parameters	2-4
2.6.2 Tuning Database Files	2-7
2.6.3 Tuning Automatic Segment-Space Management (ASSM)	2-7
2.7 Reusing Database Connections	2-8
2.8 Enabling Data Source Statement Caching	2-8
2.9 Controlling Concurrency	2-9
2.9.1 Setting Server Connection Limits	2-9
2.9.2 Configuring Connection Pools	2-11
2.9.3 Tuning the WebLogic Server Thread Pool	2-11
2.10 Setting Logging Levels	2-12
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	3-1

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 Fusion Middleware Control	4-2
4.3	Oracle WebLogic Server Administration Console	4-2
4.4	WebLogic Diagnostics Framework (WLDF).....	4-3
4.5	WebLogic Scripting Tool (WLST).....	4-3
4.6	DMS Spy Servlet.....	4-3
4.6.1	Viewing Performance Metrics Using the Spy Servlet	4-3
4.6.2	Using the DMS Spy Servlet	4-3
4.7	Native Operating System Performance Commands.....	4-5
4.8	Network Performance Monitoring Tools	4-5

5 Using the Oracle Dynamic Monitoring Service

5.1	About Dynamic Monitoring Service (DMS)	5-1
5.1.1	Understanding Common DMS Terms and Concepts	5-1
5.2	Understanding DMS Availability.....	5-6
5.3	Understanding DMS Architecture	5-6
5.4	Viewing DMS Metrics	5-7
5.4.1	Viewing Metrics Using the Spy Servlet.....	5-7
5.4.2	Viewing Metrics with WLDF (WebLogic Diagnostic Framework).....	5-8
5.4.3	Viewing metrics with WLST (Oracle WebLogic Server).....	5-8
5.4.4	Viewing metrics with JConsole.....	5-9
5.4.5	Viewing metrics with Oracle Enterprise Manager	5-10
5.5	Accessing DMS Metrics with WLDF.....	5-10
5.6	DMS Execution Context	5-11
5.6.1	DMS Execution Requests and Sub-Tasks.....	5-11
5.6.2	DMS Execution Context Usage.....	5-12
5.6.3	DMS Execution Context Communication	5-12
5.7	DMS Tracing and Events	5-13
5.7.1	Configuring the DMS Event System.....	5-13
5.7.2	Configuring Destinations	5-16
5.7.3	Understanding the Format of DMS Events in Log Messages	5-24
5.7.4	Understanding DMS Event Actions.....	5-27
5.8	DMS Best Practices.....	5-28

Part II Core Components

6 Tuning Oracle HTTP Server

6.1	About Oracle HTTP Server.....	6-1
6.2	Monitoring Oracle HTTP Server Performance	6-1
6.3	Basic Tuning Considerations	6-2
6.3.1	Tuning Oracle Server Directives.....	6-2

6.3.2	Reducing Process Availability with Persistent Connections.....	6-7
6.3.3	Logging Options for Oracle HTTP Server.....	6-8
6.4	Advanced Tuning Considerations	6-9
6.4.1	Tuning Oracle HTTP Server.....	6-9
6.4.2	Tuning Oracle HTTP Server Security	6-11

7 Tuning Oracle Metadata Service

7.1	About Oracle Metadata Services (MDS).....	7-1
7.2	Monitoring Oracle Metadata Service Performance	7-1
7.3	Basic Tuning Considerations.....	7-2
7.3.1	Tuning Database Repository.....	7-2
7.3.2	Tuning Cache Configuration	7-3
7.3.3	Purging Document Version History	7-4
7.3.4	Using Database Polling Interval for Change Detection	7-5
7.4	Advanced Tuning Considerations	7-6
7.4.1	Analyzing Performance Impact from Customization	7-6

8 Tuning Oracle Fusion Middleware Security Performance Tuning

8.1	About Security Services	8-1
8.2	Basic Tuning Considerations.....	8-2
8.3	Tuning Oracle Platform Security Services.....	8-2
8.3.1	JVM Tuning Parameters	8-2
8.3.2	JDK Tuning Parameters	8-3
8.3.3	Authentication Tuning Parameters.....	8-3
8.3.4	Authorization Tuning Properties	8-3
8.3.5	OPSS PDP Service Tuning Parameters	8-6
8.4	Oracle Web Services Security Tuning	8-8
8.4.1	Choosing the Right Policy	8-9
8.4.2	Policy Manager.....	8-9
8.4.3	Configuring the Log Assertion to Record SOAP Messages	8-9
8.4.4	Configuring Connection Pooling	8-9
8.4.5	Monitoring the Performance of Web Services.....	8-10

Part III Oracle Fusion Middleware Server Components

9 Tuning Oracle Application Development Framework (ADF)

9.1	About Oracle ADF	9-1
9.2	Basic Tuning Considerations.....	9-1
9.2.1	Oracle ADF Faces Configuration and Profiling	9-2
9.2.2	Performance Considerations for ADF Faces.....	9-2
9.2.3	Tuning ADF Faces Component Attributes	9-11
9.2.4	Performance Considerations for Table and Tree Components.....	9-13
9.2.5	Performance Considerations for autoSuggest	9-14
9.2.6	Data Delivery - Lazy versus Immediate.....	9-14
9.2.7	Performance Considerations for DVT Components.....	9-15
9.3	Advanced Tuning Considerations	9-16

9.3.1	ADF Server Performance	9-16
-------	------------------------------	------

10 Tuning Oracle TopLink

10.1	About Oracle TopLink and EclipseLink	10-1
10.2	Basic Tuning Considerations.....	10-2
10.2.1	Creating Efficient SQL Statements and Queries	10-2
10.2.2	Tuning Cache Configuration	10-9
10.2.3	Tuning the Mapping and Descriptor Configurations	10-15
10.2.4	Using Data Partitioning	10-16
10.3	Advanced Tuning Considerations	10-16
10.3.1	Integrating with Oracle Coherence	10-16
10.3.2	Analyzing EclipseLink JPA Entity Performance.....	10-16

Part IV SOA Suite Components

11 Tuning the SOA Infrastructure

11.1	About the SOA Infrastructure.....	11-1
11.2	Tuning SOA Work Managers.....	11-1
11.2.1	Configuring database connections with the SOADDataSource property.....	11-2
11.2.2	Configuring Work Managers with the SOAMaxThreadsConfig attribute.....	11-3
11.3	Tuning SOA Infrastructure Parameters.....	11-3
11.4	Using Advanced Tuning Options.....	11-5
11.4.1	Enabling the Composite Lazy Loading Feature.....	11-5
11.4.2	Changing Modularity Profiles	11-6
11.4.3	Tuning Your Database for SOA Processes.....	11-7
11.4.4	Tuning Event Delivery Network Parameters	11-13
11.4.5	Tuning the WebLogic Server.....	11-17
11.5	Advanced Tuning for Work Managers.....	11-18
11.5.1	Configuring Fair Share Request Class for SOA Work Managers	11-19
11.5.2	Creating a New Work Manager Constraint.....	11-19

12 Tuning Oracle BPEL Process Manager

12.1	About BPEL Process Manager	12-1
12.2	Tuning BPEL Parameters.....	12-1
12.2.1	Tuning BPEL Engine	12-1
12.2.2	Tuning BPEL in a Composite.....	12-4
12.3	Using Other Tuning Strategies.....	12-5
12.3.1	Identifying Tables Impacted By Instance Data Growth.....	12-5
12.3.2	Troubleshooting Frequent Transaction Timeouts While Displaying Composite Instances	12-6

13 Tuning Oracle Mediator

13.1	About Oracle Mediator	13-1
13.2	Tuning Mediator Parameters	13-1
13.2.1	Using Resequencer for Messages	13-2

14	Tuning Oracle Managed File Transfer	
14.1	About Managed File Transfer	14-1
14.2	Tuning MFT Parameters	14-1
14.2.1	Tuning Remote FTP / SFTP/ FILE Type Sources.....	14-4
14.2.2	Minimizing MDS label	14-4
14.2.3	Adjusting the Materialized Views Refresh Interval	14-5
15	Tuning Oracle Business Rules	
15.1	About Oracle Business Rules	15-1
15.2	Tuning Oracle Business Rules.....	15-1
16	Tuning Oracle Business Process Management	
16.1	About Oracle Business Process Management.....	16-1
16.2	Tuning Business Process Management Parameters.....	16-1
16.3	Using Other Tuning Strategies.....	16-3
16.3.1	Tuning Oracle Workspace Applications	16-4
16.3.2	Process Measurement.....	16-5
17	Tuning Oracle Human Workflow	
17.1	About Oracle Human Workflow	17-1
17.2	Tuning Human Workflow	17-1
17.3	Using Other Tuning Strategies.....	17-3
17.3.1	Improving Server Performance	17-3
17.3.2	Completing Workflows Faster	17-4
17.3.3	Tuning the Identity Provider	17-5
17.3.4	Tuning the Database.....	17-6
18	Tuning Oracle Business Activity Monitoring	
18.1	About Oracle Business Activity Monitoring.....	18-1
18.2	Tuning BAM Server Parameters.....	18-1
18.3	Other Tuning Strategies	18-3
18.3.1	Creating an Index Column	18-3
18.3.2	Tuning Loggers	18-3
18.3.3	Tuning Continuous Query Service.....	18-3
19	Tuning Oracle Service Bus	
19.1	About Oracle Service Bus	19-1
19.2	Tuning OSB Parameters.....	19-1
19.2.1	Tuning Oracle Service Bus with Work Managers.....	19-2
19.2.2	Tuning OSB Operation Settings.....	19-2
19.3	Using Other Tuning Strategies.....	19-5
19.3.1	Tuning Resequencer in OSB.....	19-5
19.3.2	Considering Design Time for Proxy Applications.....	19-7
19.3.3	Tuning XQuery	19-8

19.3.4	Tuning Poller-based Transports	19-9
--------	--------------------------------------	------

20 Tuning Oracle Enterprise Scheduler Service

20.1	About Enterprise Scheduler Service.....	20-1
20.2	Tuning Enterprise Scheduler Service Parameters	20-1

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:

- [Audience](#)
- [Documentation Accessibility](#)
- [Conventions](#)

Audience

Tuning Performance is aimed at a target audience of Application developers, Oracle Fusion Middleware administrators, database administrators, and Web masters.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
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"](#)

Introduction and Roadmap

This chapter gives you an overview of the Performance Tuning guide by describing the contents and organization of the chapters.

This chapter contains the following sections:

- [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

Tuning Performance is for 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, "Using the Oracle Dynamic Monitoring Service"](#) provides an overview and features available in the Oracle Dynamic Monitoring Service (DMS).
- [Chapter 6, "Tuning Oracle HTTP Server,"](#) 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 7, "Tuning Oracle Metadata Service,"](#) provides tuning tips for Oracle Metadata Service (MDS). MDS is used by Oracle Application Development Framework to manage metadata.

- [Section 8, "Tuning 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 9, "Tuning Oracle Application Development Framework \(ADF\),"](#) 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 10, "Tuning Oracle TopLink,"](#) provides some of the available performance options for Java Persistence API (JPA) entity architecture. Oracle TopLink includes EclipseLink as the JPA implementation.
- [Chapter 11, "Tuning the SOA Infrastructure,"](#) 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 12, "Tuning Oracle BPEL Process Manager"](#) 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 13, "Tuning Oracle Mediator"](#) 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 14, "Tuning Oracle Managed File Transfer,"](#) describes how to tune Oracle Managed File Transfer, a new product in 12c (12.1.3). Oracle Managed File Transfer (MFT) is a high performance, standards-based, end-to-end managed file gateway. It features design, deployment, and monitoring of file transfers using a lightweight web-based design-time console that includes file encryption, scheduling, and embedded FTP and SFTP servers.
- [Chapter 15, "Tuning Oracle Business Rules"](#) describes the technology that enables automation of business rules; it also discusses the extraction of business rules from procedural logic such as Java code or BPEL processes.
- [Chapter 16, "Tuning Oracle Business Process Management"](#) describes how to tune Oracle Business Process Management (BPM), which provides a seamless integration of all stages of the application development life cycle from design-time and implementation to run-time and application management.
- [Chapter 17, "Tuning Oracle Human Workflow"](#) 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 18, "Tuning Oracle Business Activity Monitoring"](#) 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 19, "Tuning Oracle Service Bus"](#) describes how to tune the Oracle Service Bus, which provides connectivity, routing, mediation, management and also some process orchestration capabilities. It is part of a larger system where it plays the role of an intermediary between two or more applications (servers).
- [Chapter 20, "Tuning Oracle Enterprise Scheduler Service"](#) describes how to tune the Oracle Enterprise Scheduler Service, which enables scheduling and running jobs within a particular time frame, or workshift, using rules to create work assignments.

1.3 Related Documentation

For more information, see the following documents in the Oracle Fusion Middleware 12c (12.1.3) documentation set:

- *Administering Oracle Fusion Middleware*
- *Understanding Oracle Fusion Middleware*
- *Securing Applications with Oracle Platform Security Services*
- *High Availability Guide*
- *Tuning Performance of Oracle WebLogic Server*
- *Administering Oracle HTTP Server*
- *Administering Web Services*

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 Java applications.

This chapter contains the following sections:

- [Section 2.1, "Identifying Top Performance Areas"](#)
- [Section 2.2, "Securing Sufficient Hardware Resources"](#)
- [Section 2.3, "Tuning the Operating System"](#)
- [Section 2.4, "Tuning Java Virtual Machines \(JVMs\)"](#)
- [Section 2.5, "Tuning the WebLogic Server"](#)
- [Section 2.6, "Tuning Database Parameters"](#)
- [Section 2.7, "Reusing Database Connections"](#)
- [Section 2.8, "Enabling Data Source Statement Caching"](#)
- [Section 2.9, "Controlling Concurrency"](#)
- [Section 2.10, "Setting Logging Levels"](#)

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

2.2 Securing Sufficient Hardware Resources

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 "Verifying Certification, System Requirements, and Interoperability" in *Planning an Installation of Oracle Fusion Middleware*.

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 never 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, "Controlling Concurrency"](#).

2.3 Tuning 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 Tuning Java Virtual Machines (JVMs)

How you tune your Java virtual machine (JVM) greatly affects the performance of Oracle Fusion Middleware and your applications. For more information on tuning your JVM, see "Tuning Java Virtual Machines (JVM)" in *Tuning Performance of Oracle WebLogic Server*.

2.5 Tuning the WebLogic Server

If your Oracle Fusion Middleware applications are using the WebLogic Server, see "Tuning WebLogic Server" in *Tuning Performance of Oracle WebLogic Server*.

2.6 Tuning 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.

Note: The information in this section is a subset of database tuning information for Fusion Middleware. Make sure that you have also reviewed the *Database Performance Tuning Guide*.

This section covers the following:

- [Tuning Database Parameters](#)
- [Tuning Database Files](#)
- [Tuning Automatic Segment-Space Management \(ASSM\)](#)

Note: Always review the tuning guidelines in your database-specific vendor documentation.

2.6.1 Tuning 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.

The database that is used for SOA is configured with the suggested values. Tuning the database involves adjusting the sizing parameters based on the available resource and load on the database.

The `sga_target`, `pga_aggregate_target`, and `processes` parameters from [Table 2-2](#) are examples of such parameters that need to be tuned based on SGA and PGA advisories and looking into the number of open processes during peak load.

Table 2–2 Important Oracle 12c Database Tuning Parameters

Parameter	Description	Tuning Recommendation
audit_trail Default: DB	This parameter enables or disables database auditing.	Set this parameter to NONE if there is NO policy to audit database activity. Enabling auditing can impact performance.
plsql_code_type Default: INTERPRETED	PLSQL_CODE_TYPE specifies the compilation mode for PL/SQL library units. Possible modes are as follows: <ul style="list-style-type: none"> 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. 	Set this parameter to NATIVE.
nls_sort Default: Derived from NLS_LANGUAGE	NLS_SORT specifies the collating sequence for ORDER BY queries. <ul style="list-style-type: none"> If the value is a named linguistic sort, the collating sequence is based on the order of the defined linguistic sort. Most languages supported by the NLS_LANGUAGE parameter also support a linguistic sort with the same name. If the value is set to BINARY, then the collating sequence is based on the numeric value of characters. This requires fewer system resources. 	Set this parameter to BINARY.
open_cursors Default: 50	This parameter specifies the maximum number of open cursors that a session can have at once. Open cursors are handles to private SQL areas. The value of OPEN_CURSORS should be high enough to prevent your application from running out of open cursors.	Increase this parameter to 500.
session_cached_cursors Default: 50	This parameter 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.	Increase this parameter to 500.
_b_tree_bitmap_plans Default: TRUE	This parameter enables or disables the use of bitmap access paths for b-tree indexes.	Set this parameter to FALSE.

Table 2–2 (Cont.) Important Oracle 12c Database Tuning Parameters

Parameter	Description	Tuning Recommendation
processes Default: 100	This parameter 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. The SESSIONS parameter is deduced from this value.	For most systems, increasing this parameter value to 1500 should suffice. For a large scale system, such as databases with large number of users, the recommended value is 5000.
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.	Consider setting the MEMORY_TARGET to NONE. Then set the SGA and PGA targets separately as setting MEMORY_TARGET does not allocate sufficient memory to SGA and PGA as needed.
sga_target Default: 0	Setting this parameter to a nonzero value enables Automatic Shared Memory Management. This can simplify configuration and improve performance.	For small systems, use a minimum of 2GB. For large systems, set this to 18GB.
pga_aggregate_target Default: 0	This parameter specifies the target aggregate PGA memory available to all server processes attached to the instance.	For small systems, use a minimum of 1GB. For large systems, set this to 8GB.
Disk_asynch_io Default: TRUE	DISK_ASYNCH_IO controls whether I/O to data files, control files, and log files is asynchronous. It decides what parallel server processes can overlap I/O requests with CPU processing during table scans.	If your platform supports asynchronous I/O, leave this parameter to its default value of TRUE. Otherwise, set it to FALSE.
Filesystemio_options Default: None	FILESYSTEMIO_OPTIONS specifies I/O operations for file system files.	Set this parameter to SETALL.
Secure_Files Default: PERMITTED	This parameter specifies how to store LOB objects from tables.	Set this parameter to ALWAYS.
parallel_max_servers Default: PARALLEL_THREADS_PER_CPU*CPU_COUNT*concurrent_parallel_users*5	This parameter specifies the maximum number of parallel execution processes and parallel recovery processes for an instance. As demand increases, an Oracle Database increases the number of processes from the number created at instance startup up to this value.	Set this parameter to 12.
job_queue_processes Default: 1000	Specifies the maximum number of job slaves per instance that can be created for the execution of DBMS_JOB jobs and Oracle Scheduler (DBMS_SCHEDULER) jobs.	Set this parameter to 12.
shared_servers Default: 0 (or) 1	Specifies the number of server processes that you want to create when an instance is started.	Set this parameter to 0.

In addition, you may consider setting a minimum value for SHARED_POOL_SIZE and DB_CACHE_SIZE to minimize frequent resizing.

2.6.2 Tuning Database Files

In addition to tuning the database parameters, the database administrator should properly configure the REDO logs, UNDO tablespace, and TEMP tablespaces to meet the demands of the expected or observed database workload. This is an empirical task. The recommendations in this section are intended to provide initial guidance in these areas.

The location of the database files should be optimized for I/O performance and growth. Segment Advisor should be leveraged to optimize the use of segment space and assure performance degradation does not occur. The advisor can provide historical growth trends of segments, which can be used to proactively plan for growth. See "Using the Segment Advisor" in *Oracle Database Administrator's Guide* for more information.

REDO Logs

Under demanding workloads, the size of the redo log files can influence performance. Generally, larger redo log files provide better performance. Undersized log files increase checkpoint activity and log file switches, which reduces performance. You can obtain sizing advice on the Redo Log Groups page of the Enterprise Manager.

Depending on your storage configuration and performance characteristics, you may need to redistribute redo logs to optimize I/O performance. Redo log files should be placed on a disk separate from data files to improve I/O performance.

See "Managing the Redo Log" in the *Oracle Database Administrator's Guide* for more information.

UNDO Tablespace

The suggested minimum size for the UNDO tablespace is 6 GB with auto-extend enabled. It is recommended that the default mode of automatic undo management be leveraged to maximize performance and efficiency.

The Oracle Enterprise Manager Automatic Undo Management Advisor should be leveraged to set configuration details for undo tablespace and retention settings. This advisor also provides access to the Undo Advisor that assesses the effect and provides advice of a new undo retention setting. For more information about using advisors, see "The Undo Advisor PL/SQL Interface" in the *Oracle Database Administrator's Guide*.

TEMP Tablespace

Oracle recommends the use of locally-managed temporary tablespaces with UNIFORM extents and the default size of 1 MB.

For tuning TEMP tablespaces for SOA, see [Section 11.4.3.2](#).

2.6.3 Tuning 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`.

While configuring tablespaces, consider setting the extent allocation type to `SYSTEM`. If the allocation type is set to `UNIFORM`, it might impact performance.

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

2.7 Reusing 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 *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.8 Enabling 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.

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.

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 WebLogic Server Administration Console Online Help*.

For more information on using statement caching, see "Increasing Performance with the Statement Cache" in the *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.9 Controlling 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.

- [Setting Server Connection Limits](#)
- [Configuring Connection Pools](#)
- [Tuning the WebLogic Server Thread Pool](#)

2.9.1 Setting Server 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 *Administering 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 6, "Tuning Oracle HTTP Server"](#) 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 6-1, "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 HTTP Server Modules

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 *Administering Oracle HTTP Server*.

2.9.2 Configuring Connection Pools

Connection pooling is configured and maintained per Java runtime. Connections are not shared across different runtimes. To use connection pooling, no configuration is required. Configuration is necessary only if you want to customize how pooling is done, such as to control the size of the pools and which types of connections are pooled.

You configure connection pooling by using a number of system properties at program startup time. Note that these are system properties, not environment properties and that they affect all connection pooling requests.

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 WebLogic Server Administration Console Online Help*.

"Tuning Data Source Connection Pool Options" in *Administering JDBC Data Sources for Oracle WebLogic Server*.

2.9.3 Tuning the WebLogic Server 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 *Tuning Performance of Oracle WebLogic Server*
- "Thread Management" in *Tuning Performance of Oracle WebLogic Server*
- "MDB Thread Management" in *Tuning Performance of Oracle WebLogic Server*
- "Using Work Managers to Optimize Scheduled Work" in *Administering Server Environments for Oracle WebLogic Server*
- "Avoiding and Managing Overload" in *Administering 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 WebLogic Server Administration Console Online Help*.

The work manager and thread pool metrics can also be viewed from the Oracle Fusion Middleware Control.

2.10 Setting Logging Levels

The amount of information that 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 also 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.

Each component may have specific recommendations for logging levels. See the component chapters in this book for more information.

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.

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.

- 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 the *High Availability Guide*.
- 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

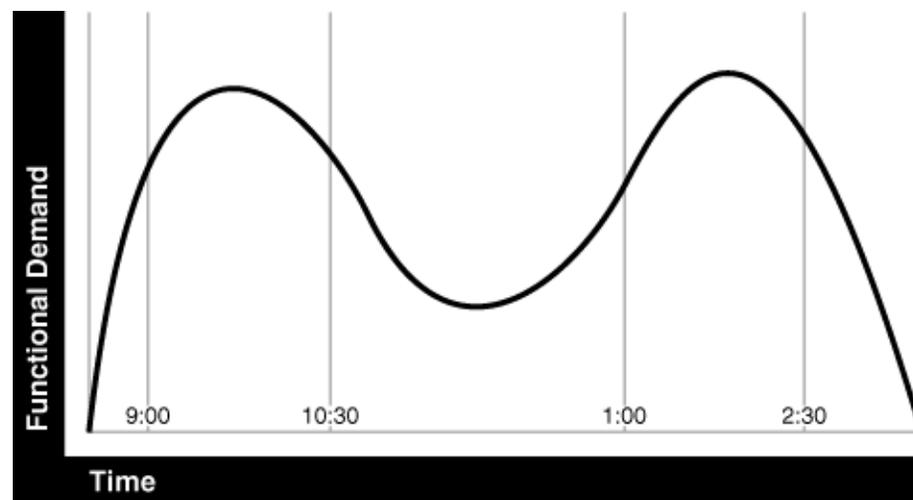
3.2.1.3 Understand User Expectations

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.

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*

Tuning Performance of Oracle WebLogic Server

Administering Oracle Fusion Middleware

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 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, "Native Operating System Performance Commands"](#)
- [Section 4.8, "Network Performance Monitoring Tools"](#)

Note: Additional monitoring information is included for most products in the product-specific chapters of this guide.

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.

Each tool is described in "Overview of Oracle Fusion Middleware Administration Tools" in *Administering Oracle Fusion Middleware*.

Note: The Oracle Process Manager and Notification Server (OPMN) is no longer used in Oracle Fusion Middleware. Instead, system components are managed by the WebLogic Management Framework, which includes WLST, Node Manager and pack and unpack. See "What Is the WebLogic Management Framework" in *Understanding Oracle Fusion Middleware*.

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 *Administering Oracle Fusion Middleware*. 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.

4.2 Oracle Enterprise Manager Fusion Middleware Control

Fusion Middleware Control is a Web browser-based, graphical user interface that you can use to monitor and administer your domain. It can manage an Oracle WebLogic Server domain with its Administration Server, one or more Managed Servers, clusters, the Oracle Fusion Middleware components that are installed, configured, and running in the domain, and the applications you deploy.

For more information, see "Getting Started Using Oracle Enterprise Manager Fusion Middleware Control" in *Administering Oracle Fusion Middleware*.

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.

For more information on using the WebLogic Server console, see "Getting Started Using Oracle WebLogic Server Administration Console" in *Administering Oracle Fusion Middleware*.

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 *Tuning Performance of Oracle 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.

For more information, see "Understanding the Diagnostic Framework" in *Administering Oracle Fusion Middleware*.

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

For more information, see "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in *Administering Oracle Fusion Middleware*.

4.6 DMS Spy Servlet

The DMS Spy servlet provides 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

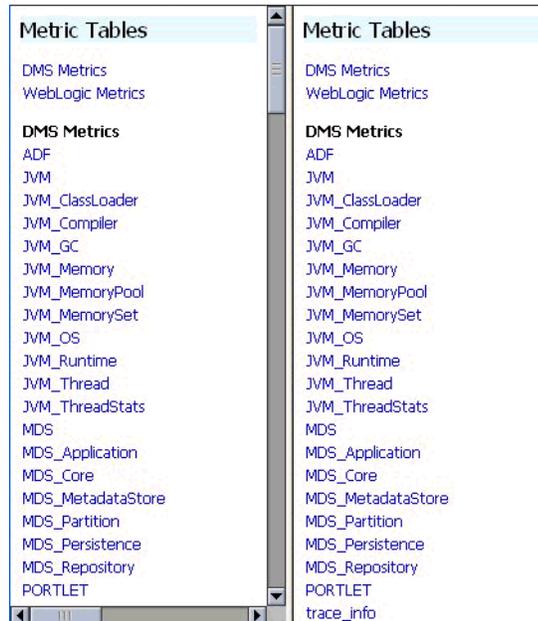
The DMS Spy Servlet is part of the DMS web application. The DMS web application's web archive file is `dms.war`, and can be found in the same directory as `dms.jar`:
`/modules/oracle.dms_12.1.2/dms.war`.

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 Java EE elements in `web.xml`.

4.6.2 Using the DMS Spy Servlet

[Figure 4–1](#) shows the initial page of the Spy servlet: both sides show the same list of metric tables.

Figure 4–1 Spy Servlet Page - Metrics Tables

Note that the Spy servlet can display metric tables for WebLogic Server and also for non-Java EE 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 4–2](#) shows the `MDS_Partition` table.

Figure 4–2 MDS Partition Table

MDS_Partition								
Name	Host	Process	readDocument	writeDocument	MDS_Application	MDS_Repository	ServerName	
oracle		WLS_Spaces:8888	active, threads avg, msec completed, ops maxActive, threads maxTime, msec minTime, msec time, msec	0 active, threads 0.106 avg, msec 254 completed, ops 1 maxActive, threads 5 maxTime, msec 0 minTime, msec 27 time, msec	0 0 0 0 0 0	webcenter(11.1.1.2.0)		WLS_Spaces
owsm		WLS_Spaces:8888	active, threads avg, msec completed, ops maxActive, threads maxTime, msec minTime, msec time, msec	0 active, threads 0 avg, msec 0 completed, ops 0 maxActive, threads 0 maxTime, msec 0 minTime, msec 0 time, msec	0 10.66 100 1 69 4 1066	wsm-pm	oracle	WLS_Spaces
webcenter		WLS_	active,	0 active,	0	webcenter(11.	mds-	WLS_

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

4.7 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.

4.8 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.

Using the Oracle Dynamic Monitoring Service

This chapter provides an overview and features available in the Oracle Dynamic Monitoring Service (DMS).

- [Section 5.1, "About Dynamic Monitoring Service \(DMS\)"](#)
- [Section 5.2, "Understanding DMS Availability"](#)
- [Section 5.3, "Understanding DMS Architecture"](#)
- [Section 5.4, "Viewing DMS Metrics"](#)
- [Section 5.5, "Accessing DMS Metrics with WLDF"](#)
- [Section 5.6, "DMS Execution Context"](#)
- [Section 5.7, "DMS Tracing and Events"](#)
- [Section 5.8, "DMS Best Practices"](#)

5.1 About Dynamic Monitoring Service (DMS)

The Oracle Dynamic Monitoring Service (DMS) enables Oracle Fusion Middleware components to provide administration tools, such as Oracle Enterprise Manager, with data regarding the component's performance, state and on-going behavior. Fusion Middleware components push data to DMS and in turn DMS publishes that data through a range of different components. DMS measures and reports metrics, trace events and system performance and provides a context correlation service for these components.

5.1.1 Understanding Common DMS Terms and Concepts

This section defines common DMS terms and concepts related to the following:

- [DMS Sensors](#)
- [DMS Nouns](#)
- [DMS Tracing and Events](#)

5.1.1.1 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 5.1.1.1.1, "DMS PhaseEvent Sensors"](#)

- [Section 5.1.1.1.2, "DMS Event Sensors"](#)
- [Section 5.1.1.1.3, "DMS State Sensors"](#)

5.1.1.1.1 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 5–1](#) lists the metrics available with PhaseEvent sensors.

Table 5–1 DMS PhaseEvent Sensor Metrics

Metric	Description
<code>sensor_name.time</code>	Specifies the total time spent in the phase <code>sensor_name</code> . Default metric: <code>time</code> is a default PhaseEvent sensor metric.
<code>sensor_name.completed</code>	Specifies the number of times the phase <code>sensor_name</code> has completed since the process was started. Optional metric
<code>sensor_name.minTime</code>	Specifies the minimum time spent in the phase <code>sensor_name</code> , for all the times the <code>sensor_name</code> phase completed. Optional metric
<code>sensor_name.maxTime</code>	Specifies the maximum time spent in the phase <code>sensor_name</code> , for all the times the <code>sensor_name</code> phase completed. Optional metric
<code>sensor_name.avg</code>	Specifies the average time spent in the phase <code>sensor_name</code> , computed as the (total time)/(number of times the phase completed). Optional metric
<code>sensor_name.active</code>	Specifies the number of threads in the phase <code>sensor_name</code> , at the time the DMS statistics are gathered (the value may change over time). Optional metric
<code>sensor_name.maxActive</code>	Specifies the maximum number of concurrent threads in the phase <code>sensor_name</code> , since the process started. Optional metric

5.1.1.1.2 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 5–2](#) describes the metric that is associated with an event sensor.

Table 5–2 DMS Event Sensor Metrics

Metric	Description
<code>sensor_name.count</code>	Specifies the number of times the event has occurred since the process started, where <code>sensor_name</code> is the name of the Event sensor as specified in the DMS instrumentation API. Default: <code>count</code> is the default metric for an event sensor. No other metrics are available for an event sensor.

5.1.1.1.3 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 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 5–3](#) 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 5–3 DMS State Sensor Metrics

Metric	Description
<code>sensor_name.value</code>	Specifies the metric value for <code>sensor_name</code> , using the type assigned when <code>sensor_name</code> is created. Default: <code>value</code> is the default State metric.
<code>sensor_name.count</code>	Specifies the number of times <code>sensor_name</code> is updated. Optional metric
<code>sensor_name.minValue</code>	Specifies the minimum value for <code>sensor_name</code> since startup. Optional metric
<code>sensor_name.maxValue</code>	Specifies the maximum value this <code>sensor_name</code> since startup. Optional metric

5.1.1.1.4 Sensor Naming Conventions The following list describes 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 5-1](#), [Table 5-2](#), and [Table 5-3](#).

5.1.1.2 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 the attribute that identifies the noun's type. Nouns that represent similar types of entities will typically have the same noun type and will usually record a common set of measurements for each of those entities.

5.1.1.2.1 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 with the namespace and localpart. The noun name is preceded by the full name of its parent, and a delimiter.

`/dmsDemo/BasicBinomial/{http://mynamespace/}JAXWSHelloService` 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.

5.1.1.2.2 General DMS Naming Conventions and Character Sets DMS names should be as compact as possible. When you define noun and sensor names, avoid special characters such as white space, slashes, periods, parenthesis, commas, and control characters.

[Table 5-4](#) shows DMS replacement for special characters in names.

Table 5-4 Replacement for Special Characters in DMS Names

Character	DMS Replacement Character
Space character	Underscore character: <code>_</code>
Period character: <code>.</code>	Underscore character: <code>_</code>
Control character	Underscore character: <code>_</code>
Less than character: <code><</code>	Open parenthesis: <code>(</code>

Table 5–4 (Cont.) Replacement for Special Characters in DMS Names

Character	DMS Replacement Character
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.

5.1.1.2.3 Noun and Noun Type Naming Conventions The following conventions are used when naming noun and noun types:

- A noun name should be unique.
- 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.
- 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.

5.1.1.3 DMS Tracing and Events

Conceptually DMS generates a stream of events; each event is in response to one of the event-producing actions being performed on the DMS API by the components that integrate with DMS (such as a sensor being updated). That stream of events can be completely ignored or routed (and optionally filtered) to destinations that can respond in some way to events.

[Table 5–5](#) provides a list of DMS tracing and event terminology.

Table 5–5 DMS Tracing and Event Terminology

DMS Term	Definition
Condition	<p>A condition is the logic behind a condition filter. It determines which events may pass through a filter, based on the rules defined in the condition. Every condition filter has zero or one root condition, but conditions may include AND or OR arguments together to create compound conditions. The single root condition can describe a relatively complex rule.</p> <p>Two types of condition exist:</p> <ul style="list-style-type: none"> ■ Noun Type Condition - operates on the name of the noun type associated with a sensor or noun event. ■ Context Condition - operates on the values currently set within the current Execution Context. <p>For more information on using conditions, see Section 5.7, "DMS Tracing and Events".</p>
Destination	<p>A destination implements a mechanism for reacting to events that are passed to it. For example, a destination could log events to a file, another could send transformed copies of event to the Java Flight Recorder, yet another might render information gleaned from incoming events as data in an MBean.</p>
Event Route	<p>An event route connects a filter to a destination. Event routes may be enabled or disabled.</p>
Filter	<p>An event tracing filter selectively passes a subset of all possible DMS runtime events. Filters can be configured with rules that determine which events are passed and which are blocked.</p> <p>For example it is possible to define filters to:</p> <ul style="list-style-type: none"> ■ Only pass sensor updates that are made when the execution context has a key-value pair of "role"-"admin" ■ Only pass sensor updates from nouns of type "JDBC_Statement" <p>For more information on using filters, see Section 5.7, "DMS Tracing and Events".</p>
Listener	<p>A DMS listener is also known as the destination. See Section 5.7.2, "Configuring Destinations" for more information.</p>

5.2 Understanding DMS Availability

DMS functionality is available on all certified Java EE servers. This includes both the runtime features and supporting commands. Also, several features of DMS will operate in JSE applications and standalone C applications.

For more information on which servers are certified, see the Oracle Fusion Middleware Certification Matrix.

5.3 Understanding DMS Architecture

DMS consists of the following features:

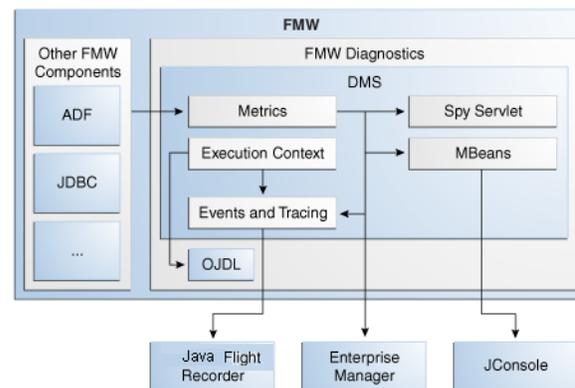
- **DMS Metrics** - The DMS metrics feature provides Java and C APIs that are used by Oracle Fusion Middleware components for instrumenting code with performance measurements and other useful state metrics.
- **Execution Context** - Execution Context supports the maintenance and propagation of a specific context structure throughout the Oracle stack. By exploiting the propagated context structure Oracle FMW components can record diagnostic information (such as log records) that can be correlated between different components and products running on the same or different servers and hosts. For

more information see [Section 5.6, "DMS Execution Context"](#).

- **Events and Tracing** - Event Tracing enables you to configure live tracing with no restarts. DMS metrics updated during the course of using Oracle Fusion Middleware products may be traced using the DMS Event Tracing feature. The system has been designed to facilitate not only tracing, but also to support other functionality that may be driven from DMS activity.

[Figure 5–1](#) shows the components of DMS and how they interact with other Oracle Fusion Middleware components. Arrows show the direction in which information flows from one component to the next.

Figure 5–1 DMS Interactions with Oracle Fusion Middleware Components



5.4 Viewing DMS Metrics

Oracle Fusion Middleware components are instrumented with DMS metrics in order to collect information that developers, system administrators, and support analysts can use to analyze system performance or monitor system status. The Fusion Middleware Control online help provides information on each of the specific metrics. See "Viewing the Performance of Oracle Fusion Middleware" in *Administering Oracle Fusion Middleware* for information on accessing metric information.

The Oracle Fusion Middleware metrics come from various sources and locations. They include MBean attributes and DMS metrics. They also come from non-Java EE servers, such as Oracle servers.

The following sections describe how to use various tools to view the DMS metrics:

- [Viewing Metrics Using the Spy Servlet](#)
- [Viewing Metrics with WLDF \(WebLogic Diagnostic Framework\)](#)
- [Viewing metrics with WLST \(Oracle WebLogic Server\)](#)
- [Viewing metrics with JConsole](#)
- [Viewing metrics with Oracle Enterprise Manager](#)

5.4.1 Viewing Metrics Using the Spy Servlet

The Spy Servlet is part of the DMS Application that is deployed by default on JRF-extended installations. The Spy Servlet is launched from

`http://<host>:<port>/dms/Spy`. The default port for WebLogic is 1521.

The DMS Application's web archive file is `dms.war`, and can be found in the same directory as `dms.jar`: `oracle_common/modules/oracle.dms_12.1.2/dms.war`.

For more information see [Section 4.6, "DMS Spy Servlet"](#).

Note: The Spy Servlet is secured using standard Java EE declarative security in the web-application's `web.xml` file, and will only grant access to the Spy Servlet to members of the Administrator's group.

5.4.2 Viewing Metrics with WLDF (WebLogic Diagnostic Framework)

You can use WebLogic Diagnostic Framework (WLDF) to harvest DMS metrics from DMS metric MBeans. You can also use WLDF to monitor changes to the attribute value of an MBean. For more information see "Configuring the Harvester for Metric Collection" in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

5.4.3 Viewing metrics with WLST (Oracle WebLogic Server)

DMS provides three commands to view metrics in WLST:

Use this command...	To do this...
<code>displayMetricTableNames()</code>	<p>List the names of the available metric tables.</p> <p>If you have a large number of DMS metric tables, consider using the <code>outputfile</code> parameter with <code>displayMetricTableNames()</code>. This is useful when the output is expected to be large. When <code>displayMetricTableNames()</code> has the <code>outputfile</code> parameter, it returns null to the script instead of the whole output. This prevents the command from running out of memory.</p> <p>NOTE: In 12c, the command syntax for <code>displayMetricTableNames()</code> differs slightly for system components (such as OHS). After you connect WLST to Node Manager using <code>nmConnect()</code> command, you must specify both server name and server type explicitly.</p> <p>For example:</p> <pre>displayMetricTableNames(servertype="OHS", servers="ohs1")</pre>

Use this command...	To do this...
<code>displayMetricTables()</code>	<p>Show the content of the DMS metric tables.</p> <p>If you have a large number of DMS metric tables, consider using the <code>outputfile</code> parameter with <code>displayMetricTables()</code>. This is useful when the output is expected to be large. When <code>displayMetricTables()</code> has the <code>outputfile</code> parameter, it returns null to the script instead of the whole output. This prevents the command from running out of memory.</p> <p>NOTE: In 12c, the command syntax for <code>displayMetricTables()</code> differs slightly for system components (such as OHS). After you connect WLST to Node Manager using <code>nmConnect()</code> command, you must specify both server name and server type explicitly.</p> <p>For example:</p> <pre>displayMetricTables(servertype="OHS", servers="ohs1")</pre>
<code>dumpMetrics()</code>	<p>Display metrics in the internal format. Valid formats for the <code>dumpMetrics</code> command include raw, xml and pdml.</p> <p>If you have a large number of DMS metric tables, consider using the <code>outputfile</code> parameter with <code>dumpMetrics()</code>. This is useful when the output is expected to be large. When <code>dumpMetrics()</code> has the <code>outputfile</code> parameter, it returns null to the script instead of the whole output. This prevents the command from running out of memory.</p> <p>NOTE: In 12c, the command syntax for <code>dumpMetrics()</code> differs slightly for system components (such as OHS). After you connect WLST to Node Manager using <code>nmConnect()</code> command, you must specify both server name and server type explicitly.</p> <p>For example:</p> <pre>dumpMetrics() (servertype="OHS", servers="ohs1")</pre>

As well as displaying textual output, these commands also return a structured object or single value that you can use in a script to process.

For more information on using these commands, see the following:

- "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in *Administering Oracle Fusion Middleware*
- "DMS Metric Commands" in *WLST Command Reference for Infrastructure Components*

5.4.4 Viewing metrics with JConsole

To provide a standards-based way to access metrics, DMS exposes them through MBeans. An MBean will be created and registered for each typed noun with the runtime MBean Server. The DMS sensors contained by the noun are exposed as the attributes of the MBean. Exposing the DMS metrics as MBeans allows administrators

to use tools such as JConsole (the Java monitoring and management console), and other Java Management Extension (JMX) clients, to access the DMS metrics.

MBeans also allow for integration with other Oracle diagnostics software such as WLDF (WebLogic Diagnostics Framework), which is described in [Section 5.5](#). The noun name and noun type are exposed as the name and type properties of the metric MBean object name. The MBean domain name is "oracle.dms". The object name also reflects the DMS noun hierarchy.

Note: You can use JConsole to view DMS generated MBeans on a Java EE server either locally or remotely. DMS generates an MBean for each Java DMS noun that has a valid noun type. It does not generate MBeans for the non-Java EE component's metrics and the DMS nouns that have no noun types. Each DMS metric contained under the noun is mapped to an attribute in the metric MBean.

5.4.5 Viewing metrics with Oracle Enterprise Manager

Oracle Fusion Middleware automatically and continuously measures data regarding the component's performance, state and on-going behavior. The metrics are automatically enabled; there is no need to set options or perform any extra configuration to collect them. For more information see [Section 4.2, "Oracle Enterprise Manager Fusion Middleware Control"](#).

5.5 Accessing DMS Metrics with WLDF

The WebLogic Diagnostics Framework (WLDF) provides a diagnostic feature that allows MBean attributes to be harvested and monitored for specific conditions. This provides a proactive way of monitoring activity in your environment and creating E-mail and JMX notifications when a condition is triggered.

The following steps describe how to configure WLDF to send an E-mail notification using the WebLogic Administration Console:

1. Select an existing or create a new Diagnostics Module from the Diagnostics screen.
2. Click on the **Watches and Notifications** tab.
3. Click **New**.
4. Enter a Watch Name and click **Next**.
5. Enter the text as the Watch Rule and click **Next**.

```
(${ServerRuntime//[NOUNTYPE]oracle.dms:name=/starWars/alliance,type=NounType//forceBalance_value} = 'BAD')
```

6. Select **Use a manual reset alarm** and click **Next**. The manual reset option means that once an E-mail is triggered, you must reset the watch using the WebLogic Administration Console.
7. Select the E-mail notification type and click **Finish**.

It is also possible to configure WLDF to collect the MBean data for offline storage and analysis. This is achieved by configuring a WLDF Diagnostic Module to collect specific MBean attributes, and can be done so using the WebLogic Administration Console.

For more information on using WLDF to harvest and monitor MBean data see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

5.6 DMS Execution Context

The DMS execution context is the mechanism by which requests (such as or RMI requests) can be uniquely identified and thus tracked as they flow through the system. It also provides a means by which context information can be communicated between cooperating Fusion Middleware components involved in fulfilling requests.

5.6.1 DMS Execution Requests and Sub-Tasks

The DMS execution context has been developed with the understanding that a single request (or task) may form the root of a tree of sub-tasks that are coordinated to complete the request or root task. Consider the following examples of requests and their associated sub-tasks:

1. An request sent directly to Oracle WebLogic Server from a browser:
 - Root task only on Oracle WebLogic Server
2. An request sent through Oracle Server (acting as a reverse proxy) to Oracle WebLogic Server:
 - Root task on Oracle Server
 - Single sub-task on Oracle WebLogic Server
3. An request sent from Oracle Server (acting as a reverse proxy) to Oracle WebLogic Server that then requires invocation of two remote web services from Oracle WebLogic Server in order to fulfill the request:
 - Root task on Oracle Server
 - Single sub-task on Oracle WebLogic Server
 - Two sub-sub-tasks, one on each web service

A DMS execution context is composed of the following:

- A unique identifier, the ECID

The Execution Context ID (ECID) is unique for each new root task and is shared across the tree of tasks associated with the root task.
- A relationship identifier, the RID

The Relationship ID (RID) is an ordered set of numbers that describes the location of each task in the tree of tasks. The leading number is usually a zero. A leading number of 1 indicates that it has not been possible to track the location of the sub-task within the overall sub-task tree.
- A set of name-value pairs by which globally relevant data can be shared among Oracle Fusion Middleware components.

The following three scenarios illustrate how ECID and RID are used when an request is sent from Oracle Server (acting as a reverse proxy) to an Oracle WebLogic Server and the server requires invocation of two remote web services from Oracle WebLogic Server.

1. Root task on Oracle Server:
 - New ECID = B5C094FA...BE4AE8
 - Root RID = 0
2. Single sub-task on Oracle WebLogic Server:
 - Same ECID = B5C094FA...BE4AE8

- Sub-task RID = 0:1
- 3. Two Sub-tasks, one on each web service:
 - First web service invoked
 - Same ECID = B5C094FA...BE4AE8
 - Sub-task RID = 0:1:1
 - Second web service invoked
 - Same ECID = B5C094FA...BE4AE8
 - Sub-task RID = 0:1:2

5.6.2 DMS Execution Context Usage

The most immediate benefits of the DMS execution context are realized when attempting to correlate log messages between servers. The Oracle standard format for logging involves a field dedicated to the ECID. Once the ECID is known, when its read from an ERROR level log message for example, it is possible to locate all other log messages associated with that task by querying the log files for messages containing that ECID.

The following example shows a very specific case of using the command:

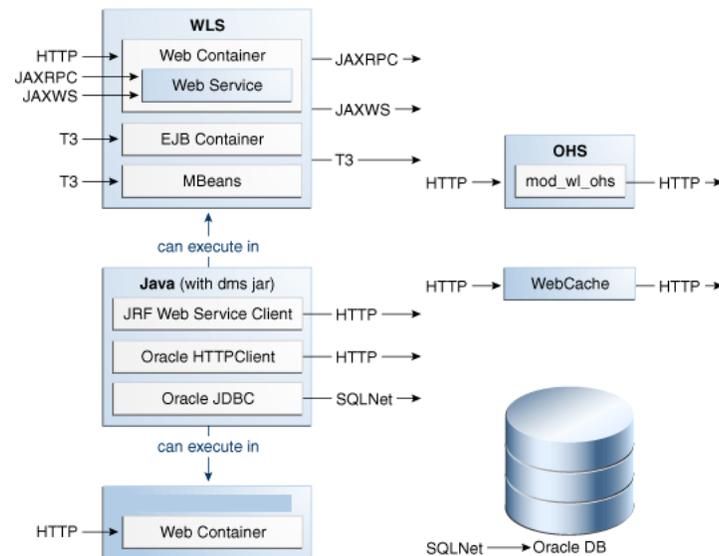
```
displayLogs (ecid="B5C094FA...BE4AE8" );
```

In this example, any log files with messages that contain the ECID B5C094FA...BE4AE8 will be displayed.

5.6.3 DMS Execution Context Communication

Figure 5–2 shows the components that cooperate in order to communicate the DMS execution context between each other. Arrows pointing to a component indicate the protocols that are inspected for incoming context information. Outgoing arrows show protocols to which context information is added. It is possible for a single component to send requests to itself, passing context information in that request.

Figure 5–2 DMS Execution Context Communication Protocols



5.7 DMS Tracing and Events

DMS can selectively trace the following:

- DMS sensor lifecycle events (create, update, delete of state sensors, event sensors and phase sensors)
- Context events (start, stop)
- events (start, stop)

The configuration that controls which of these types of events are traced, and how those events are processed, is recorded in the `dms_config.xml` file. The DMS trace configuration is split into three parts:

1. Filter Configuration

Defines the rules that select the events that are of interest

2. Destination Configuration

Defines how the events are used

3. eventRoute Configuration

Defines which filters are wired to which destinations

A filter can be associated with one or more destinations thus granting the administrator the ability to define a filter rule once and have the resulting subset of all possible events processed on one or more different destinations.

The configuration can be modified using the DMS configuration MBean or WLST commands at runtime; this makes the DMS tracing feature invaluable for diagnosing issues within a specific time period or collecting specific data at a specific time for a specific set of criteria.

For more information, see "Configuring Selective Tracing Using WLST" in *Administering Oracle Fusion Middleware*.

The following types of filter rules are supported:

- Event Type Conditions
Used to identify if an event was triggered from the START or STOP of a PHASE_SENSOR
- Context Type Conditions
Used to identify if the event was generated from a unit of work whose context contains a value (for example, "USER")
- Noun Type Conditions
Used to identify if the event was triggered from a sensor whose noun is of a specific type (for example, JDBC_CONNECTION)
- Logical AND and OR combinations of the above conditions

5.7.1 Configuring the DMS Event System

Configuration is recorded in each server's `dms_config.xml` file. MBean updates can be made at runtime using command line interface (CLI) commands and through the Event Configuration Mbean. Configuration updates are applied to the running system in a thread safe, but non-atomic, manner.

The object name of the DMS Event configuration MBean is:

```
oracle.dms.event.config:name=DMSEventConfigMBean,type=JMSEventConfig
```

To review the current state of your system's DMS event configuration, use the following command:

```
listDMSEventConfiguration([server=<server>])
```

The resulting output will look similar to this:

```
Event routes:
    FILTER      : auto662515911
    DESTINATION : destination1
    ENABLED     : true
    FILTER      : filter0
    DESTINATION : q
    ENABLED     : true
Filters with no event route:
    Fred

Destinations with no event route:
    des4
```

5.7.1.1 Adding and Editing Filters

Filters define the rules that select which events are considered for tracing.

The following example shows how to add a filter that selects all events related to JDBC operations:

```
addDMSEventFilter(id='myJDBCFilter', props={'condition': 'NOUNTYPE sw JDBC_'})
```

Or:

```
addDMSEventFilter(id='myJDBCFilter', props={'condition': 'NOUNTYPE startsWith
JDBC_'})
```

This filter assumes that all DMS sensor updates associated with JDBC operations are performed on nouns of types whose names begin "JDBC_".

If the rule must be modified, the filter may be updated as shown in the following example:

```
updateDMSEventFilter(id="myJDBCFilter", props={'condition': 'NOUNTYPE startsWith
JDBC_ OR NOUNTYPE startsWith MDS_'});
```

As of Oracle Fusion Middleware 11.1.1.6.0, the following shortened convenience operators have been added. Operators can be specified using either the shortened or longer name.

Note that operators with an underscore have been deprecated in favor of the ODL format, which is to use mixed case. For example, `not_equals` becomes `notEquals` or `ne`. The old format will still work, but is discouraged.

Noun Type Operators

equals, eq	notEquals, ne
contains	in
startsWith, sw	

Context Operators

equals, eq	notequals, ne
isnull	isnotnull
startswith, sw	contains
lt	gt

Example:

```
addDMSEventFilter(id='mdsbruce', name='MyFilter', props={'condition':
'NOUNTYPE eq MDS_Connections AND CONTEXT user ne bruce'})
```

```
addDMSEventFilter(id='mdsbruce', name='MyFilter', props={'condition':
'NOUNTYPE equals MDS_Connections AND CONTEXT user notequals bruce'})
```

For more information about the syntax used to describe a filter's rule (the condition property), refer to the WebLogic Scripting Tool Command Reference or the command help.

5.7.1.2 Adding and Editing Destinations

Destinations encapsulate logic for responding to events. For example, a basic destination will log the event, a different destination may transform an event and pass it to another system for further processing.

The following example shows how to add a destination that will log events:

```
addDMSEventDestination(id="myLoggerDestination",
class="oracle.dms.trace2.runtime.LoggerDestination",
props={"loggerName": "myLogger"});
```

Note that merely adding the destination is not sufficient for events to be logged; to log the events, you must associate a filter with a destination using an eventRoute, and the eventRoute must be enabled (default).

The types of destination available, and their configuration options, are described in [Section 5.7.2](#). The following example shows how to edit an existing destination:

```
updateDMSEventDestination(id="myLoggerDestination",
props={"loggerName": "myTraceLogger"});
```

5.7.1.3 Adding and Editing Event Routes

The following example shows how to join the filter and destination created above:

```
addDMSEventRoute(filterid='myJDBCFilter', destinationid='myLoggerDestination')
```

Note that you can invoke `addDMSEventRoute` without an explicit `filterId`. In these scenarios, all events are passed to the destination without filtering.

To remove a filter or destination, you must first remove the event routes associated with the filter or destination (even if the event route is disabled). For example, if you wanted to remove `myJDBCFilter`, you would first need to remove the eventRoute created in the previous example, and then remove the filter as shown in the following example:

```
removeDMSEventRoute(filterid='myJDBCFilter', destinationid='myLoggerDestination')
removeDMSEventFilter(id='myJDBCFilter')
```

5.7.1.4 Compound Operations

It is possible to create a filter and an eventRoute based on that filter using a single command (rather than using two separate commands as shown in [Section 5.7.1.3](#)). Note, however, that the destination to be used by the event route must already be defined:

```
enableDMSEventTrace (destinationid='myLoggerDestination', condition='NOUNTYPE
starts_with JDBC_')
```

In the example above, `enableDMSEventTrace` automatically creates a filter with the specified condition, and also creates and enables an event route using the new filter and the nominated destination. The output is shown in the following example:

```
Filter "auto605449842" using Destination "myLoggerDestination" added, and
event-route enabled for server "AdminServer"
```

5.7.2 Configuring Destinations

DMS offers the following types of destinations:

- [LoggerDestination](#)
- [MBean Creator Destination](#)
- [Request Tracker Destination](#)
- [Java Flight Recorder Destination](#)

5.7.2.1 LoggerDestination

Description	The <code>LoggerDestination</code> writes each event to the associated logger.
Implementing Class	<code>oracle.dms.trace2.runtime.LoggerDestination</code>
Properties	
<code>loggerName</code>	The name of the ODL logger to which events will be written.

Instances of logger destinations write events to the named logger at a log level of `FINER`.

The `loggerName` property specifies the name of a logger, but the logger does not necessarily have to be described in `logging.xml`, though it can be. If the logger name refers to a logger that is explicitly named in `logging.xml`, then the logger is referred to as a static logger (see [Section 5.7.2.1.1](#)). If the logger name refers to a logger that is not explicitly named in `logging.xml`, then the logger is referred to as a dynamic logger (see [Section 5.7.2.1.2](#)).

Use in the default configuration: the default configuration defines a logger destination, with an identification of `LoggerDestination`. This particular instance does not form part of any eventRoute and therefore is not active. It is provided for convenience, and uses a dynamic logger.

5.7.2.1.1 Static Loggers and Handlers Loggers are the objects to which log records are presented. Log handlers are the objects through which log records are written to log files.

For complete control over the log file to which DMS trace data is written, define the logger named in the logger destination in `logging.xml`. Doing this allows you to

explicitly define the name of the log file, the maximum size, format, file rotation and policies.

Oracle recommends using commands (like the example below) to update the configuration.

```
setLogLevel(logger="myTraceLogger", level="FINER", addLogger=1);

configureLogHandler(name="my-trace-handler", addToLogger=["myTraceLogger"],
path="/tmp/myTraceLogFiles/trace", maxFileSize="10m", maxLogSize="50m",
handlerType="oracle.core.ojdl.logging.ODLHandlerFactory", addHandler=1,
useParentHandlers=0);

configureLogHandler(name="my-trace-handler",
propertyName="useSourceClassandMethod", propertyValue="false", addProperty=1);
```

For more information on logging configuration, see "Managing Log Files and Diagnostic Data" in the *Administering Oracle Fusion Middleware*.

The use of the optional property `useSourceClassandMethod` set to `FALSE` prevents the 'SRC_CLASS' and 'SRC_METHOD' from appearing in every message and will marginally improve performance by reducing file output times.

For static loggers, consider setting the `useParentHandlers` parameter to `FALSE`, otherwise duplicate event messages will be logged to `[server]-diagnostics.log`, and shown in a log query.

See [Section 5.7.3, "Understanding the Format of DMS Events in Log Messages"](#) for more information about interpreting logger output.

5.7.2.1.2 Dynamic Loggers and Handlers If the named logger has no associated handler defined in `logging.xml`, then the logger destination will dynamically create a handler object that will write to a file in the server's default log output directory. (Instances of logger destinations write events to the named logger at a log level of `FINER`.) The file name will be the logger's name followed by `-event.log`. For instance, in the example in [Section 5.7.2.1.1](#), DMS events would be written to `myTraceLogger-event.log`.

5.7.2.1.3 Default Locations of the logging.xml File The `logging.xml` file can typically be found in one of the following platform locations:

Platform	Server	Location
Oracle WebLogic Server	AdminServer	ORACLE_HOME/WLS_Home/user_projects/domains/base_domain/config/fmwconfig/servers/AdminServer/logging.xml

5.7.2.1.4 Using a CLI Command to Query the Trace Log File If the logger destination's logger and handler are defined in `logging.xml` then you can take advantage of the `displayLogs()` command to conveniently access logged trace data without having to manually locate or search for it.

Examples:

- To display all the log messages for the `myTraceLogger`:

```
displayLogs(query='MODULE equals myTraceLogger')
```
- To display only the log messages for `myTraceLogger` which have an ECID of `'0000HpmSpLWEkJQ6ub3FEH194kwB000004'`:

```
displayLogs(query='MODULE equals myTraceLogger and ECID equals
0000HpmSpLWEkJQ6ub3FEH194kwB000004')
```

- To display only the log messages for myTraceLogger which have an ECID of '0000HpmSpLWEkJQ6ub3FEH194kwB000004' in the last 10 minutes:

```
displayLogs(query='MODULE equals myTraceLogger and ECID equals
0000HpmSpLWEkJQ6ub3FEH194kwB000004', last=10)
```

- To display all the log messages from a dynamic logger the log's file name must be included:

```
displayLogs(disconnected=1, log=DOMAIN_
ROOT+"/servers/AdminServer/logs/myTraceLogger-event.log")
```

5.7.2.2 MBean Creator Destination

Description	The MBean creator destination make nouns accessible as MBeans, exposing their metrics as attributes, for access via WLDF, JConsole, etc.
Implementing Class	oracle.dms.jmx.MetricMBeanFactory

Use in the default configuration: An instance of the MBean Creator destination is configured and active by default, and will create MBeans for all nouns created in the server.

By associating an instance of this destination type with a filter based on a noun-type rule, it is possible to expose (as MBeans) only those noun types that are of interest to the administrator.

Although it is possible to modify the configuration associated with an MBean creator destination at runtime, it must be understood that the reinitialization process for this type of destination may impact performance. Frequent runtime reconfiguration is therefore discouraged.

Note that WebLogic Diagnostic Framework (WLDF) can be used to harvest DMS metrics exposed by the MBean creator destination. For more information about WLDF, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

5.7.2.2.1 Metric MBean Object Name The noun name and noun type are exposed as the name and type properties of the metric MBean object name. The MBean domain name is "oracle.dms". The object name also reflects the DMS noun hierarchy.

For example if the noun's full path name is:

```
/oracle/dfw/ofm/base_domain/AdminServer
```

and the noun type is DFW_Incident, the object name of the MBean representing the noun is

```
oracle.dms:Location=AdminServer,name=/oracle/dfw/ofm/base_
domain/AdminServer,type=DFW_Incident.
```

5.7.2.3 Request Tracker Destination

Description	The Request Tracker destinations maintains a list of active requests, and makes the requests accessible to other Diagnostic Framework (DFW) components.
--------------------	---

Implementing Class	oracle.dms.event.RequestTrackerDestination
Properties	
excludeHeaderNames	Comma separated list of header names to exclude from tracking

Use in the default configuration: An instance of the request tracker destination is enabled by default. In the case of a DFW incident being generated the active request list will be dumped automatically, allowing an administrator to correlate the failure with a specific request.

For each request the following information will be dumped:

- Uniform Resource Identifier (URI)
- Start time of the request
- Execution Context ID (ECID)
- Query string
- Headers

When the request tracker is not enabled the Request Dump will output the following:

```
Requests are not being tracked. To enable request tracking enable the DMS
oracle.dms.event.RequestTrackerDestination in dms_config.xml
```

5.7.2.3.1 Executing the Request Tracker Dump The information being maintained by the request tracker can be accessed manually. In order to execute the dump that reports the request information the WLST `executeDump` command can be used, when connected to a server, as follows:

```
> executeDump(name=".requests")
Active Requests:

StartTime: 2009-12-14 02:24:41.870
ECID: 0000IMChyqEC8xT6uBF9EH1B9X9^000009,0
URI: /myApp/Welcome.jsp
QueryString:
Headers:
  Host: myHost.myDomain.com:7001
  Connection: keep-alive
  User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.5
(KHTML, like Gecko) Chrome/4.0.249.30 Safari/532.5
  Accept:
application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*
/*;q=0.5
  Accept-Encoding: gzip,deflate
  Cookie: ORA_MOS_LOCALE=en%7CGB; s_nr...
  Accept-Language: en-GB,en-US;q=0.8,en;q=0.6
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

5.7.2.4 Java Flight Recorder Destination

The Java Flight Recorder (JFR) records information regarding the runtime status and behavior of the Java JVM. JFR also exposes an API through which third party events can be reported.

By themselves DMS traces and JFR traces only show part of the picture of the actions being performed in the server. DMS integration with JFR enhances the diagnostic information available to administrators and developers as follows:

1. Application level events and JVM level events can be reported as a single sequence therefore avoiding the need to combine such events from separate log files based only on timestamp (which may not tick over fast enough to accurately order events created at or around the same time).
2. Recent DMS activity can be dumped, retroactively, from the JVM at will.
3. Recent DMS and JVM events can be dumped to disk in the event of a fatal error that causes the JVM to exit gracefully.
4. The DMS ECID can be used to correlate activity relating to the same request, or unit of work, across the span of a JFR recording.
5. The DMS ECID can be used to collect diagnostic information from all systems involved with an event, or series of events, recorded by JFR.

5.7.2.4.1 Dynamically Derived JFR Event Types – Names, Values and Descriptions A DMS noun type will be associated with a JFR InstantEvent event type:

- The name of the JFR event type for a noun type will be the noun type's name with the suffix "state".
- The path of the JFR event type for a noun type will be "dms/" followed by the producer-name, followed by the event type name.
- Event sensors will not contribute any values to the noun type's JFR event type.
- The values of the JFR event for a noun type are described in [Table 5–6](#):

Table 5–6 Values of the JFR Event for a Noun Type

Value Name	Description	Relational	Notes
ECID	The Execution Context ID (ECID) associated with the action.	Yes	
RID	The RID associated with the action.	Yes	
<noun type> name	The full path of the noun.		This field will be populated with the full path of the noun. The field's name assumes that the noun_type meaningfully categorizes all objects being measured by the nouns of that type.
<state-sensor-name>	The value of the state sensor.	No	Each state sensor belonging to the noun will contribute one of these values to the instant event. There may be more than one value in each noun.
event name	The name of the event sensor that was updated, left null otherwise.	No	The event name field is required for being able to count the number of times a DMS event sensor has been updated in a recording (event sensors do not contribute values to an event type).

A DMS phase sensor will be associated with a JFR DurationEvent event type:

- The name of the JFR event type for a phase sensor belonging to a noun of a particular noun type will be the noun type's name following by the phase sensor's name.
- The path of the JFR event for a noun type will be "dms/" followed by the producer-name, followed by the event type name.
- The values of the duration event will be as above (except for the sensorName value). For example the "stop" of a phase event will result in a JFR duration event being reported to JFR that contains the state information of the phase event's parent noun.

Several DMS objects allow integrators to add descriptions. Descriptions from DMS objects will be used as follows:

- Noun type description will be used in creation of the JFR event type
- State and event sensor descriptions will not be applied – there is nowhere to apply them.
- Phase sensor descriptions will be applied to their JFR event type.

5.7.2.4.2 Examples of Dynamically Derived Producers and Events [Table 5-7](#) provides examples for the rules described in [Section 5.7.2.4.1](#):

Table 5-7 Examples of Dynamically Derived Producers and Events

DMS	Java Flight Recorder (JFR)
<p>Noun type: JDBC_Connection</p>	<p>Producer Name: JDBC The Producer Name is based on the leading component of the noun path.</p>
<p>Noun path: /JDBC/Driver/CONNECTION_7</p>	<p>Event Type 1 Event Type Name: JDBC_Connection State <noun type> State Event Type Path: dms/JDBC/JDBC_Connection_State dms/<leading component of noun path>/<noun type>/_State</p>
<p>Sensors: CreateStatement (P) CreateNewStatement (P) DBWaitTime (P) JDBC_Connection_Url (S) JDBC_Connection_Username (S)</p>	<p>Fields:</p> <ul style="list-style-type: none"> ▪ ECID ▪ RID ▪ JDBC_Connection name Value will be the full path of the noun ▪ JDBC_Connection_Url Value will be that of the state sensor of this name at the time of the event ▪ JDBC_Connection_Username Value will be that of the state sensor of this name at the time of the event ▪ Event Name Value will be one of the following: <ul style="list-style-type: none"> ▪ The name of the DMS event sensor whose activation caused this JFR event instance ▪ Null if this JFR event instance was created for a state sensor update
<p>Where: P: Phase Sensor S: State Sensor E: Event Sensor</p>	

Table 5–7 (Cont.) Examples of Dynamically Derived Producers and Events

DMS	Java Flight Recorder (JFR)
	Producer Name: JDBC Event Type 2 Event Type Name: JDBC_Connection CreateStatement Event Type Path: dms/JDBC/JDBC_Connection_CreateStatement Fields: <ul style="list-style-type: none"> ▪ ECID ▪ RID ▪ JDBC_Connection name ▪ JDBC_Connection_Url ▪ JDBC_Connection_Username
	Producer Name: JDBC Event Type 3 Event Type Name: JDBC_Connection CreateNewStatement Event Type Path: dms/JDBC/JDBC_Connection_CreateNewStatement Fields: <ul style="list-style-type: none"> ▪ ECID ▪ RID ▪ JDBC_Connection name ▪ JDBC_Connection_Url ▪ JDBC_Connection_Username
	Producer Name: JDBC Event Type 4 Event Type Name: JDBC_Connection DBWaitTime Event Type Path: dms/JDBC/JDBC_Connection_DBWaitTime Fields: <ul style="list-style-type: none"> ▪ ECID ▪ RID ▪ JDBC_Connection name ▪ JDBC_Connection_Url ▪ JDBC_Connection_Username

5.7.3 Understanding the Format of DMS Events in Log Messages

Table 5–8 describes the fields that make up a DMS event. Field elements are separated by ":" (with a few exceptions). Sample events are provided to illustrate the position of the field within an actual event string.

Table 5–8 Event Formatting Descriptions

Applicable Events	Field Number	Name	Description
All	1	Version number	The version number of the event format For example: v1:1280737384058:_REQUEST:STOP:/MyWebApp/emp
All	2	Event time	The time at which the event occurred For example: v1:1280737384058:_REQUEST:STOP:/MyWebApp/emp
All	3	Source object type	The type of object on which an action was performed to produce the event including: <ul style="list-style-type: none"> ■ NOUN ■ EVENT_SENSOR ■ STATE_SENSOR ■ PHASE_SENSOR ■ EXECUTION_CONTEXT ■ _REQUEST For example: v1:1280737384058:_REQUEST:STOP:/MyWebApp/emp
All	4	Action type	The type of action that resulted in the generation of this event. A given source object type may not necessarily produce events for every action type: <ul style="list-style-type: none"> ■ CREATE ■ UPDATE ■ DELETE ■ START ■ STOP ■ ABORT For example: v1:1280737384058:_REQUEST:STOP:/MyWebApp/emp
Nouns	5	Noun type	The name of the noun type For example: v1:1281344803506:NOUN:CREATE:JDBC_ Connection:/JDBC/JDBC Data Source-0/CONNECTION_1
	6	Noun path	The full path identifying the noun to which the sensor belongs For example: v1:1281344803506:NOUN:CREATE:JDBC_ Connection:/JDBC/JDBC Data Source-0/CONNECTION_1

Table 5–8 (Cont.) Event Formatting Descriptions

Applicable Events	Field Number	Name	Description
All Sensor Types	5	Noun type	The name of the noun type to which this sensor belongs For example: v1:1280503318973:STATE_SENSOR:UPDATE:JDBC_Connection:LogicalConnection:/JDBC/JDBC Data Source-0/CONNECTION_1:State.ANY:LogicalConnection@13bed086
	6	Sensor name	The name of the sensor For example: v1:1280737383069:PHASE_SENSOR:STOP:JDBC_Connection:DBWaitTime:/JDBC/JDBC Data Source-0/CONNECTION_1:1280737382950:1280737383069
	7	Noun path	The full path identifying the noun to which the sensor belongs For example: v1:1280737383069:PHASE_SENSOR:STOP:JDBC_Connection:DBWaitTime:/JDBC/JDBC Data Source-0/CONNECTION_1:1280737382950:1280737383069
Phase Sensor Types	8	Start token	The start token of the phase. For example: v1:1280737383069:PHASE_SENSOR:STOP:JDBC_Connection:DBWaitTime:/JDBC/JDBC Data Source-0/CONNECTION_1:1280737382950:1280737383069
	9	Stop token	The end token of the phase. For example: v1:1280737383069:PHASE_SENSOR:STOP:JDBC_Connection:DBWaitTime:/JDBC/JDBC Data Source-0/CONNECTION_1:1280737382950:1280737383069

Table 5–8 (Cont.) Event Formatting Descriptions

Applicable Events	Field Number	Name	Description
State Sensor Types	8	State value type	The type of value held by the state sensor including: <ul style="list-style-type: none"> State.DOUBLE State.INTEGER State.LONG State.OBJECT State.ANY For example: v1:1280503318973:STATE_SENSOR:UPDATE:JDBC_Connection:LogicalConnection:/JDBC/JDBC Data Source-0/CONNECTION_1:State.ANY:LogicalConnection@13bed086
	9	State value	The value of the state represented in string form. For example: v1:1280503318973:STATE_SENSOR:UPDATE:JDBC_Connection:LogicalConnection:/JDBC/JDBC Data Source-0/CONNECTION_1:State.ANY:LogicalConnection@13bed086
Requests	5	URI	Uniform Resource Identifier (URI) identifies the resource upon which to apply the request. For example: v1:1280737382889:_REQUEST:START:/myWebApp/showEmployees v1:1280737384058:_REQUEST:STOP:/myWebApp/showEmployees
Execution Context	5	ECID,RID	The context identifier (composed of ECID and RID separated by a comma). For execution context events the complete substring starting at the first character after the fourth event field separator (":") records the ECID,RID identifiers - the context identifiers may contain ":" but these should not be interpreted as event field separators. For example: v1:1280737384058:EXECUTION_CONTEXT:STOP:bc4fd0668f79d507:367c127f:12a23f2013c:-8000-00000000000000f73,0

5.7.4 Understanding DMS Event Actions

Table 5–9 shows the action types that can be performed on source object types.

Table 5–9 Actions Performed on Source Object Types

	Create	Update	Delete	Start	Stop	Abort
Noun	Yes	-	Yes	-	-	-
Event Sensor	Yes	Yes	Yes	-	-	-
Phase Sensor	Yes	-	Yes	Yes	Yes	Yes
State Sensor	Yes	Yes	Yes	-	-	-

Table 5–9 (Cont.) Actions Performed on Source Object Types

	Create	Update	Delete	Start	Stop	Abort
Execution Context	-	-	-	Yes	Yes	-
Request	-	-	-	Yes	Yes	-

5.8 DMS Best Practices

The use of DMS metrics can have an impact on application performance. When adding metrics, consider the following:

- Use 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.

System clocks are not necessarily as accurate as their precision implies. For example, a system clock that reports time in milliseconds may not tick (change) once per millisecond. Instead, it may take up to 15ms to tick as shown in the following example:

Table 5–10 Default System Clock Time versus Actual Time (in milliseconds)

Actual Time	System Time
12:00:00.000	12:00:00.000
12:00:00.001	12:00:00.000
12:00:00.002	12:00:00.000
[...]	
12:00:00.014	12:00:00.000
12:00:00.015	12:00:00.015
12:00:00.016	12:00:00.015

[Table 5–10](#) shows a phase with a 12ms duration that runs from actual time 12:00:00.002 to 12:00:00.014 would be calculated in system time as having a duration of zero. Similarly, a phase with a 2ms duration running from 12:00:00.014 to 12:00:00.016 would be reported in system time as having a duration of 15ms.

Note: These behaviors are more evident on some operating systems than others. Use caution when analyzing individual periods of time that are shorter than the tick period of the system clock. Configuring DMS to use a higher resolution clock will cause DMS to record phase sensor activations with higher resolution, but the accuracy will still be limited by the underlying system.

- Configure DMS Clocks for Reporting Time for Java

Selecting the high resolution clock changes clocks for all applications running on the server 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 5–11](#) shows supported values for the `oracle.dms.clock` property.

[Table 5–12](#) shows supported values for the `oracle.dms.clock.units` property.

Table 5–11 *oracle.dms.clock Property Values*

Value	Description
DEFAULT	Specifies that DMS use the default clock. With the default clock, DMS uses the Java call <code>java.lang.System.currentTimeMillis</code> to obtain times for <code>PhaseEvents</code> . The default value for the units for the default clock is <code>MSECS</code> .
HIGHRES	The Java Highres clock uses <code>System.nanoTime()</code> (no JNI required).

Table 5–12 *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 the clock property is set with a value not listed in [Table 5–11](#), DMS uses the default clock. If the `oracle.dms.clock` property is not set, DMS uses the default clock.

- When the clock units property is set to a value not listed in [Table 5-12](#), DMS uses the default units for the specified clock.

Part II

Core Components

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

- [Chapter 5, "Using the Oracle Dynamic Monitoring Service"](#)
- [Chapter 6, "Tuning Oracle HTTP Server"](#)
- [Chapter 7, "Tuning Oracle Metadata Service"](#)

Note: For information on performance tuning the Oracle WebLogic Server, see *Tuning Performance of Oracle WebLogic Server*.

Tuning Oracle HTTP Server

This chapter discusses the techniques for optimizing Oracle HTTP Server performance.

This chapter contains the following sections:

- [Section 6.1, "About Oracle HTTP Server"](#)
- [Section 6.2, "Monitoring Oracle HTTP Server Performance"](#)
- [Section 6.3, "Basic Tuning Considerations"](#)
- [Section 6.4, "Advanced Tuning Considerations"](#)

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.

6.1 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 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 Server.

For more information see "Introduction to Oracle Server" *Administering 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/>.

6.2 Monitoring Oracle HTTP Server Performance

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 Server, see "Managing and Monitoring Server Processes" in *Administering Oracle HTTP Server*.

For monitoring, Oracle HTTP Server uses the 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 minimum, maximum, and 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.

For more information on using these DMS metrics, see [Section 5.4.3, "Viewing metrics with WLST \(Oracle WebLogic Server\)"](#).

6.3 Basic Tuning Considerations

The following tuning configurations may improve the performance of the Oracle HTTP Server. Always consult your own use case scenarios to determine if these settings are applicable to your deployment.

- [Tuning Oracle Server Directives](#)
- [Reducing Process Availability with Persistent Connections](#)
- [Logging Options for Oracle HTTP Server](#)

6.3.1 Tuning Oracle Server Directives

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

More information on configuring the Oracle Server, see "Management Tools for Oracle Server" in *Administering 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/OHSCComponent/<ohsname>/httpd.conf` file.

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 10g Release 3 documentation at

<http://www.oracle.com/technology/documentation/appserver10132.html>.

Table 6–1 Oracle HTTP Server Configuration Properties

Directive	Description
<p><code>ListenBackLog</code></p> <p>This directive maps to the Maximum Queue Length field on the Performance Directives screen.</p>	<p>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.</p> <p>Default Value: 511</p>
<p><code>MaxClients</code></p> <p>This directive maps to the Maximum Requests field on the Performance Directives screen.</p> <p>Note that this parameter is not available in <code>mod_winnt</code> (Microsoft Windows). <code>Winnt</code> uses a single process, multi-threaded model and is controlled by <code>ThreadLimit</code> directive.</p>	<p>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 <code>ListenBackLog</code> directive (after the queue of pending connections is full, new requests generate connection errors until a thread becomes available).</p> <p>You can configure the <code>MaxClients</code> directive in the <code>httpd.conf</code> file up to a maximum of 8000 (8K) (the default value is 150). If your system is not resource-saturated and you have a user population of more than 150 concurrent <code>/Thread</code> connections, you can improve your performance by increasing <code>MaxClients</code> to increase server concurrency. Increase <code>MaxClients</code> until your system becomes fully utilized (85% is a good threshold).</p> <p>Conversely, when system resources are saturated, increasing <code>MaxClients</code> does not improve performance. In this case, the <code>MaxClients</code> value could be reduced as a throttle on the number of concurrent requests on the server.</p> <p>If the server handles persistent connections, then it may require sufficient concurrent <code>d</code> or thread server processes to handle both active and idle connections. When you specify <code>MaxClients</code> to act as a throttle for system concurrency, you must consider that persistent idle <code>d</code> connections also consume <code>d/thread</code> processes. Specifically, the number of connections includes the currently active persistent and non-persistent connections and the idle persistent connections. A persistent <code>KeepAlive</code> connection consumes an <code>d</code> child process, or thread, for the duration of the connection, even if no requests are currently being processed for the connection.</p> <p>If you have sufficient capacity, <code>KeepAlive</code> should be enabled; using persistent connections improves performance and prevents wasting CPU resources reestablishing connections. Normally, you should not change <code>KeepAlive</code> parameters.</p> <p>The maximum allowed value for <code>MaxClients</code> is 8192 (8K).</p> <p>Default Value: 150</p>
<p><code>StartServers</code></p> <p>This directive maps to the Initial Child Server Processes field on the Performance Directives screen.</p>	<p>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.</p> <p>Note that the following parameters are inter-related and applicable only on UNIX platforms (<code>worker_mpm</code>):</p> <ul style="list-style-type: none"> ■ <code>MaxClients</code> ■ <code>MaxSpareThreads</code> and <code>MinSpareThreads</code> ■ <code>ServerLimit</code> and <code>StartServers</code> <p>On the Windows platform (<code>mpm_winnt</code>), as well as UNIX platforms, the following parameters are important to tune:</p> <ul style="list-style-type: none"> ■ <code>ThreadLimit</code> ■ <code>ThreadsPerChild</code> <p>Note that each child process has a set of child threads defined for them and that can actually handle the requests. Use <code>ThreadsPerChild</code> in connection with this directive.</p> <p>The values of <code>ThreadLimit</code>, <code>ServerLimit</code>, and <code>MaxClients</code> can indirectly affect this value. Read the notes for these directives and use them in conjunction with this directive.</p> <p>Default Value: 2</p>

Table 6–1 (Cont.) Oracle HTTP Server Configuration Properties

Directive	Description
<p><code>ServerLimit</code></p> <p>Note that this parameter is not available in <code>mod_winnt</code> (Microsoft Windows). <code>Winnt</code> uses a single process, multi-threaded model</p>	<p>Specifies an upper limit on the number of server (child) processes that can exist or be created. This value overrides the <code>StartServers</code> value if that value is greater than the <code>ServerLimit</code> value. This is used to control the maximum number of server processes that can be created.</p> <p>Default Value: 16</p>
<code>ThreadLimit</code>	<p>Specifies the upper limit on the number of threads that can be created under a server (child) process. This value overrides the <code>ThreadsPerChild</code> value if that value is greater than the <code>ThreadLimit</code> value. This is used to control the maximum number of threads created per process to avoid conflicts/issues.</p> <p>Default Values:</p> <ul style="list-style-type: none"> ■ Windows Multi-Processing Module (<code>mpm_winnt</code>): 1920 ■ All others: 64
<p><code>ThreadsPerChild</code></p> <p>This directive maps to the Threads Per Child Server Process field on the Performance Directives screen.</p>	<p>Sets the number of threads created by each server (child) process at startup.</p> <p>Default Value: 150 when <code>mpm_winnt</code> is used and 25 when Worker MPM is used.</p> <p>The <code>ThreadsPerChild</code> directive works with other directives, as follows:</p> <p>At startup, Oracle HTTP Server creates a parent process, which creates several child (server) processes as defined by the <code>StartServers</code> directive. Each server process creates several threads (server/worker), as specified in <code>ThreadsPerChild</code>, and a listener thread which listens for requests and transfers the control to the worker/server threads.</p> <p>After startup, based on load conditions, the number of server processes and server threads (children of server processes) in the system are controlled by <code>MinSpareThreads</code> (minimum number of idle threads in the system) and <code>MaxSpareThreads</code> (maximum number of idle threads in the system). If the number of idle threads in the system is more than <code>MaxSpareThreads</code>, 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 <code>MinSpareThreads</code>, it creates new threads and processes if the <code>ThreadsPerChild</code> value has already been reached in the running processes.</p> <p>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.</p> <ul style="list-style-type: none"> ■ <code>ServerLimit</code> - Defines the upper limit on the number of servers that can be created. This affects <code>MaxClients</code> and <code>StartServers</code>. ■ <code>ThreadLimit</code> - Defines the upper limit on <code>ThreadsPerChild</code>. If <code>ThreadsPerChild</code> is greater than <code>ThreadLimit</code>, then it is automatically trimmed to the latter value. ■ <code>MaxClients</code> - 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 <code>ThreadsPerChild</code>. If <code>MaxClients</code> is greater than <code>ServerLimit</code> multiplied by <code>ThreadsPerChild</code>, it is automatically be trimmed to the latter value.

Table 6–1 (Cont.) Oracle HTTP Server Configuration Properties

Directive	Description
<p>MaxRequestsPerChild</p> <p>This directive maps to the Max Requests Per Child Server Process field on the Performance Directives screen.</p>	<p>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.</p> <p>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.</p> <p>Default Value: 0</p> <p>Note: On Windows systems MaxRequestsPerChild should always be set to 0 (unlimited) since there is only one server process.</p>
<p>MaxSpareThreads</p> <p>MinSpareThreads</p> <p>These directives map to the Maximum Idle Threads and Minimum Idle Threads fields on the Performance Directives screen.</p> <p>Note that these parameters are not available in mod_winnt (Windows platform).</p>	<p>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.</p> <p>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.</p> <p>Default Values:</p> <p>MaxSpareThreads: 75</p> <p>MinSpareThreads: 25</p>
<p>Timeout</p> <p>This directive maps to the Request Timeout field on the Performance Directives screen.</p>	<p>The number of seconds before incoming receives and outgoing sends time out.</p> <p>Default Value: 300</p>
<p>KeepAlive</p> <p>This directive maps to the Multiple Requests Per Connection field on the Performance Directives screen.</p>	<p>Whether or not to allow persistent connections (more than one request per connection). Set to Off to deactivate.</p> <p>Default Value: On</p>

Table 6–1 (Cont.) Oracle HTTP Server Configuration Properties

Directive	Description
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: 5 seconds
limit ulimit	Number of objects that a program uses to read or write to an open file or open network sockets. A lack of available file descriptors can impact operating system performance. Tuning the file descriptor limit can be accomplished by configuring the hard limit (<code>ulimit</code>) in a shell script which starts the OHS. Once the hard limit has been set the OHS will then adjust the soft limit (<code>limit</code>) to match. Note that configuring file descriptor limits is platform specific. Refer to your operating system documentation for more information.

6.3.2 Reducing Process Availability with Persistent Connections

If your browser supports persistent connections, you can support them on the server using the `KeepAlive` directives in the Oracle 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 5
```

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.

6.3.3 Logging Options for Oracle HTTP Server

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

6.3.3.1 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.

6.3.3.2 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 `/Apache/Apache/bin/` directory.

6.3.3.3 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_ossso` or `mod_plsql`, there is a performance cost associated with setting higher debugging levels, such as the `debug` level.

6.4 Advanced Tuning Considerations

This section provides advanced tuning recommendations which may or may not apply to your environment. Review the following recommendations to determine if the changes would improve your Server performance.

- [Tuning Oracle HTTP Server](#)
- [Tuning Oracle HTTP Server Security](#)

6.4.1 Tuning Oracle HTTP Server

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

- [Analyzing Static Versus Dynamic Requests](#)
- [Managing PL/SQL Requests](#)
- [Limiting the Number of Enabled Modules](#)
- [Monitoring Oracle HTTP Server Performance](#)

6.4.1.1 Analyzing 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.

6.4.1.2 Managing PL/SQL Requests

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
```

6.4.1.3 Limiting the Number of Enabled 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.

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, enable only those modules that are required by your deployed applications to save CPU time.

6.4.1.4 Tuning the File Descriptor Limit

A lack of available file descriptors can cause a wide variety of symptoms which are not always easily traced back to the operating system's file descriptor limit. Tuning the file descriptor limit can be accomplished by configuring the operating system's hard limit for the user who starts the OHS. Once configured, the OHS will adjust the soft limit to match the operating system limit.

Configuring file descriptor limits is platform-specific. Refer to your operating system documentation for more information. The following code example shows the command for Linux:

```
APACHECTL_ULIMIT=ulimit -S -n `ulimit -H -n`
```

Note that this limit must be reconfigured after applying a patch set.

6.4.2 Tuning Oracle HTTP Server Security

This section covers the following topics:

- [Tuning Oracle HTTP Server Secure Sockets Layer \(SSL\)](#)
- [Tuning Oracle HTTP Server Port Tunneling](#)

6.4.2.1 Tuning Oracle HTTP Server Secure Sockets Layer (SSL)

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, 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 6.4.2.1.1, "Caching SSL on Oracle HTTP Server"](#)
- [Section 6.4.2.1.2, "Using SSL Application Level Data Encryption"](#)
- [Section 6.4.2.1.3, "Tuning SSL Performance"](#)

See Also: *Securing Applications with Oracle Platform Security Services*

6.4.2.1.1 Caching SSL on Oracle HTTP Server 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 persistent connections. Oracle HTTP Server uses persistent connections by default, assuming they are supported on the client side. In over SSL as implemented by Oracle HTTP Server, SSL session state is kept in memory while the associated 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 connection). For more information see [Section 6.3.2, "Reducing Process Availability with Persistent Connections"](#).

6.4.2.1.2 Using 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 Server negotiates a cipher suite with a client based on the SSLCipherSuite attribute specified in ssl.conf. OHS 11g 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: *Administering Oracle HTTP Server* for information on using supported cipher suites.

6.4.2.1.3 Tuning SSL Performance 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 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.

6.4.2.2 Tuning Oracle HTTP Server Port Tunneling

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: *Administering Oracle HTTP Server* for information on configuring OracleAS Port Tunneling

Tuning Oracle Metadata Service

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

- [Section 7.1, "About Oracle Metadata Services \(MDS\)"](#)
- [Section 7.2, "Monitoring Oracle Metadata Service Performance"](#)
- [Section 7.3, "Basic Tuning Considerations"](#)
- [Section 7.4, "Advanced Tuning Considerations"](#)

7.1 About Oracle Metadata Services (MDS)

Oracle Metadata Services (MDS) is an application server and Oracle relational database that keeps metadata in these areas: the ClassPath, the ServletContext, database repository and in some cases, the file system. One of the primary uses of MDS is to store customizations and persisted personalization for Oracle applications. MDS is used by components such as 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 Services configuration parameters are immutable and cannot be changed at run time unless otherwise specified.

Tuning MDS tablespace and cache size is very important before tuning Oracle B2B and other Oracle products. If you are using the User's Guide for Oracle B2B to tune B2B, make sure you have completed the tuning in this chapter first.

7.2 Monitoring Oracle Metadata Service Performance

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.

7.3 Basic Tuning Considerations

Tuning is the adjustment of parameters to improve performance. The default MDS configuration must be tuned in almost all deployments. Please review the requirements and recommendations in this section carefully.

7.3.1 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:

- [Collecting Schema Statistics](#)
- [Increasing Redo Log Size](#)
- [Reclaiming Disk Space](#)
- [Monitoring the Database Performance](#)

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

7.3.1.1 Collecting 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);
```

If the performance does not improve after statistics collection, then try to flush the database shared pool to clear out the existing SQL plans by using the following command:

```
alter system flush shared_pool;
```

In general, the database should be configured with automatic statistics recollection. For additional information on gathering statistics, see "Automatic Performance Statistics" in *Oracle Database Performance Tuning Guide*.

7.3.1.2 Increasing 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 can reduce performance.

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

7.3.1.3 Reclaiming 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*.

7.3.1.4 Monitoring 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*.

7.3.2 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 *Administering Oracle Fusion Middleware*.

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 7.2, "Monitoring Oracle Metadata Service Performance"](#).

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.

7.3.2.1 Enabling 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 7.2](#)) should be less than 1. If not, consider increasing the document cache size.

7.3.3 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.

7.3.3.1 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 will be executed every *T* seconds and will remove versions that are older than the specified time *T* (in seconds). For more information, see "Changing MDS Configuration Attributes for Deployed Applications" in *Administering Oracle Fusion Middleware*.

7.3.3.2 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 *Administering Oracle Fusion Middleware*.

7.3.4 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 *Administering Oracle Fusion Middleware*.

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.

7.4 Advanced Tuning Considerations

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.

7.4.1 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 (Session) until the user logs out.

For more information about customization concepts, writing customization classes, and configuring customization classes, see "Customizing Applications with MDS" in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Tuning 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 8.1, "About Security Services"](#)
- [Section 8.2, "Basic Tuning Considerations"](#)
- [Section 8.3, "Tuning Oracle Platform Security Services"](#)
- [Section 8.4, "Oracle Web Services Security Tuning"](#)

8.1 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 *Administering Web Services* before tuning any security parameters.

8.2 Basic Tuning Considerations

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.

8.3 Tuning Oracle Platform Security Services

This section provides the following basic tuning configurations for Oracle Platform Security Services (OPSS):

- [JVM Tuning Parameters](#)
- [Authentication Tuning Parameters](#)
- [Authorization Tuning Properties](#)
- [OPSS PDP Service Tuning Parameters](#)

8.3.1 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, "Tuning Java Virtual Machines \(JVMs\)"](#).

8.3.2 JDK Tuning Parameters

Starting with Java Development Kit 7 (JDK 7), the default keystore size is now 2048 bits. JDK 6 and earlier had a default size of 1024 bits.

When using the Java keytool to generate keystores, the `-keysize` parameter can be used to control the keystore size. Larger keystores provide stronger security, though at the cost of decreased security performance. Consider your environment's use case scenarios to determine if increasing the keystores would negatively impact your security or performance thresholds.

For more information see the JDK 7 release notes at <http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html>

8.3.3 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.

8.3.4 Authorization Tuning Properties

The following Java system properties can be used to optimize authorization:

Table 8–1 Authorization Properties

Java System Properties	Default Value	Valid Values	Notes
-Djps.subject.cache.key	4	3 4 5	<p>JPS uses a Subject Resolver to convert a platform subject to JpsSubject which contains user/enterprise-role information, as well as ApplicationRole information. This information is represented as principals in the subject.</p> <p>This conversion can be CPU intensive, especially if the subject's principal set has a large population. To improve performance, JPS code caches the conversion between Platform subject and JpsSubject. Note that two subjects could be confused if their contents are the same, but the case of the principals' name is different.</p> <p>The following settings can be used to configure the cache key:</p> <ul style="list-style-type: none"> ■ 3: Use the platform subject directly as the key. Note: On WLS if the <code>principalEqualCaseinsensitive</code> flag is enabled, two subjects could be confused if their contents are the same, but the case of the principals is different. ■ 4: This setting is similar to '3' but overcomes the case-sensitive issue. This is the out-of-the-box setting. ■ 5: Instead of using the whole subject as the key, this settings uses a subset of the principal set inside the subject as the key (actually use principals of <code>WLSUserImpl</code> type). <p>This setting will accelerate the cache retrieval operation if the subject has a large principal set. On a non WLS platform (such as WAS and JBOSS, this reverts back to case '4'), so this setting is for WLS only. For this case, there is also a Time To Live setting (TTL) flag which controls how long the cache is valid, as explained below.</p>

Table 8–1 (Cont.) Authorization Properties

Java System Properties	Default Value	Valid Values	Notes
<code>-Djps.subject.cache.ttl</code>	60000ms		<p>Cache's Time To Live (TTL) for case '5' (above). This system property controls how long the cache is valid. When the time expired, the cached value is dumped. The setting can be controlled by the flag of <code>-Djps.subject.cache.ttl=xxxx</code>, where 'xxx' is the duration in milliseconds.</p> <p>Consider setting the duration of this TTL setting to the same value as the value used for the group and user cache TTL in WLS LDAP authenticator.</p>
<code>-Djps.combiner.optimize=true</code>	True	True False	<p>This system property is used to cache the protection domains for a given subject. Setting <code>-Djps.combiner.optimize=true</code> can improve Java authorization performance.</p>
<code>-Djps.combiner.optimize.lazyeval=true</code>	True	True False	<p>This system property is used to evaluate a subject's protection domain when a checkPermission occurs. Setting <code>-Djps.combiner.optimize.lazyeval=true</code> can improve Java authorization performance.</p>
<code>-Djps.policystore.hybrid.mode=true</code>	True	True False	<p>This 'hybrid mode' property is used to facilitate transition from SUN <code>java.security.Policy</code> to OPSS Java Policy Provider.</p> <p>The OPSS Java Policy Provider reads from both <code>java.policy</code> and <code>system-jazn-data.xml</code>. "Hybrid" mode can be disabled by setting the system property <code>jps.policystore.hybrid.mode</code> to false when starting the WebLogic Server. Setting <code>-Djps.policystore.hybrid.mode=false</code> can reduce runtime overhead.</p>
<code>-Djps.authz=ACC</code>	ACC	ACC SM	<p>Delegates the call to JDK API <code>AccessController.checkPermission</code> which can reduce the performance impact at run time or while debugging.</p> <p>ACC: delegate to <code>AccessController.checkPermission</code></p> <p>SM: delegate to <code>SecurityManager</code> if <code>SecurityManager</code> is set.</p>

8.3.5 OPSS PDP Service Tuning Parameters

Table 8–2 provides OPSS tuning parameters for policy store:

Table 8–2 OPSS PDP Service Tuning Parameters

Parameter	Default Value	Valid Values	Notes
oracle.security.jps.policystore.rolemember.cache.type	STATIC	STATIC, SOFT, WEAK	<p>This parameter specifies the type of role member cache. Valid only in Java EE applications.</p> <p>Valid values:</p> <ul style="list-style-type: none"> ■ STATIC: Cache objects are statically cached and can be cleaned explicitly only according to the applied cache strategy, such as FIFO. The garbage collector does not clean a cache of this type. ■ SOFT: The cleaning of a cache of this type relies on the garbage collector when there is a memory crunch. ■ WEAK: The behavior of a cache of this type is similar to a cache of type SOFT, but the garbage collector cleans it more frequently. <p>Consider maintaining the default value for the best performance.</p>
oracle.security.jps.policystore.rolemember.cache.strategy	FIFO	FIFO NONE	<p>The type of strategy used in the role member cache. Valid only in Java EE applications.</p> <p>Valid values:</p> <ul style="list-style-type: none"> ■ FIFO: The cache implements the first-in-first-out strategy. ■ NONE: All entries in the cache grow until a refresh or reboot occurs; there is no control over the size of the cache; not recommended but typically efficient when the policy footprint is very small. <p>Consider maintaining the default value for the best performance.</p>
oracle.security.jps.policystore.rolemember.cache.size	1000		<p>The size of the role member cache. The role being referred to is the enterprise role (group). You can find out the number of the groups you have in your ID store first. Then, based on your performance requirement, you can set this number to the number of the groups - full cache scenario. Or you can change to a certain percentage of the number of the groups - partial group cache scenario.</p>
oracle.security.jps.policystore.policy.lazy.load.enable	True	True False	<p>Enables or disables the policy lazy loading. If this parameter is set to false, the server initial startup time will take longer - especially in a large policy store. For faster start-up time, the recommended value is true.</p>

Table 8–2 (Cont.) OPSS PDP Service Tuning Parameters

Parameter	Default Value	Valid Values	Notes
oracle.security.jps.policystore.policy.cache.strategy	PERMISSION_FIFO	PERMISSION_FIFO NONE	The type of strategy used in the permission cache. Valid only in Java EE applications. Valid Values: <ul style="list-style-type: none"> PERMISSION_FIFO: The cache implements the first-in-first-out strategy. NONE: All entries in the cache grow until a refresh or reboot occurs; there is no control over the size of the cache; not recommended but typically efficient when the policy footprint is very small. Consider using the default value for the best performance.
oracle.security.jps.policystore.policy.cache.size	1000		The size of the permission cache. If you cache all policies, then you can set this value to the total number of grants.
oracle.security.jps.policystore.cache.updatable	True	True False	This property is used for refresh enabling. Consider maintaining the default value for the best performance.
oracle.security.jps.policystore.refresh.enable	True	True False	This property is used for refresh enabling. Consider maintaining the default value for performance.
oracle.security.jps.policystore.refresh.purge.timeout	4320000		The time, in milliseconds, after which the policy store is refreshed. Consider maintaining the default value for the best performance.
oracle.security.jps.ldap.policystore.refresh.interval	600000 (10 minutes)		The interval, in milliseconds, at which the policy store is polled for changes. Consider maintaining the default value for the best performance. This property is valid in Java EE and J2SE applications.
oracle.security.jps.policystore.rolemember.cache.warmup.enable	False	True False	This property controls the way the ApplicationRole membership cache is created. If set to True, the cache is created at server startup; otherwise, it is created on demand (lazy loading). Set to True when the number of users and groups is significantly higher than the number of application roles; set to False otherwise, that is, when the number of application roles is very high.

8.4 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](#)
- [Policy Manager](#)
- [Configuring the Log Assertion to Record SOAP Messages](#)
- [Monitoring the Performance of Web Services](#)

8.4.1 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 "Determining Which Predefined Policies to Use" in *Securing Web Services and Managing Policies with Oracle Web Services Manager* to determine which security policies are required for a deployment.

8.4.2 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.

8.4.3 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.

8.4.4 Configuring Connection Pooling

When you request that a Context instance use connection pooling by using the "com.sun.jndi.ldap.connect.pool" environment property, the connection that is used might or might not be pooled. The default rule is that plain (non-SSL) connections that use simple or no authentication are allowed to be pooled. You can

change this default to include SSL connections and the DIGEST-MD5 authentication type by using system properties. To allow both plain and SSL connections to be pooled, set the "com.sun.jndi.ldap.connect.pool.protocol" system property to the string "plain ssl" as shown below:

```
"-Dcom.sun.jndi.ldap.connect.pool.protocol="plain ssl"
```

8.4.5 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 "Overview of Performance Monitoring, Auditing, and Tuning" in *Administering Web Services*.

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 9, "Tuning Oracle Application Development Framework \(ADF\)"](#)
- [Chapter 10, "Tuning Oracle TopLink"](#)

Tuning Oracle Application Development Framework (ADF)

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 9.1, "About Oracle ADF"](#)
- [Section 9.2, "Basic Tuning Considerations"](#)
- [Section 9.3, "Advanced Tuning Considerations"](#)

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

- *Developing Fusion Web Applications with Oracle Application Development Framework*
- *Developing Web User Interfaces with Oracle ADF Faces*

9.1 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 *Developing Fusion Web Applications with Oracle Application Development Framework*.

9.2 Basic Tuning Considerations

Before building, configuring, and deploying ADF applications, review the following tuning recommendations 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](#)

9.2.1 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 9–1](#) describes some of these configuration options.

Table 9–1 ADF Configuration Options

Parameter	Description
Compression View State <code>org.apache.myfaces.trinidad.COMPRESS_VIEW_STATE</code>	Controls whether or not the page state is compressed. Latency can be reduced if the size of the data is compressed. This parameter should be set to <code>True</code> .
Enhanced Debug <code>org.apache.myfaces.trinidad.resource.DEBUG</code>	Controls whether output should be enhanced for debugging or not. This parameter should be removed or set to <code>False</code> .
Check File Modification <code>oracle.adf.view.rich.CHECK_FILE_MODIFICATION</code>	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 <code>False</code> .
Client State Method <code>oracle.adf.view.rich.CLIENT_STATE_METHOD</code>	Specifies which type of saving (<code>all</code> or <code>token</code>) should be used when client-side state saving is enabled. The default value is <code>token</code> .
Client Side Log Level <code>oracle.adf.view.rich.LOGGER_LEVEL</code>	Sets the log level on the client side. The default value is <code>OFF</code> . This parameter should be removed or set to <code>False</code> .
Assertion Processing <code>oracle.adf.view.rich.ASSERT_ENABLED</code>	Specifies when to process assertions on the client side. The default value is <code>OFF</code> . This parameter should be removed or set to <code>False</code> .

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/>.

9.2.2 Performance Considerations for ADF Faces

[Table 9–2](#) provides configuration recommendations that may improve performance of ADF Faces:

Table 9–2 Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Avoid inline JavaScript in pages.	<p>Inline JavaScript can increase response payload size, will never be cached in browser, and can block browser rendering. Instead of using inline JavaScript, consider putting all scripts in .js files in JavaScript libraries and add scripts to the page using af:resource tag.</p> <p>TIP: Consider using af:resource rather than trh:script when possible.</p>
Configure the JSP timeout parameter.	<p>Using the JavaServer Pages (JSP) timeout parameter causes infrequently used pages to be flushed from the cache by the following setting in web.xml:</p> <pre><servlet> <servlet-name> oraclejsp <init-param> <param-name> jsp_timeout </param-name> <param-value> x </param-value> </init-param> </servlet-name> </servlet></pre> <p>NOTE: Set parameter <i>x</i> based on your own use case scenarios.</p>
Create a single toolbar item with a drop-down popup.	<p>When the browser size is small because of the screen resolution, the menubar/toolbar overflow logic becomes expensive in Internet Explorer 7 and 8. It especially has problems with laying out DOM structures with input fields.</p> <p>Create a single toolbar item with a drop-down popup and put all the input fields inside it. This popup should have deferred child creation and contentDelivery="lazy".</p>
Remove unknown rowCount.	<p>A table that has an unknown rowCount can impact performance because getting the last set of rows takes excessive scrolling from the user and the application can appear to be very slow.</p> <p>Remove unknown rowCount by setting DeferEstimatedRowCountProperty="false" on the view object (VO).</p>
Disable pop-ups that cannot be displayed by the user.	<p>The fnd:attachment component, when stamped in a table, can generate an excessive amount of DOM and client component. The amount of DOM + Client component is ~8K per cell which impacts the performance of the entire page especially on slower browsers.</p> <p>Most cells have no attachments initially and only one popup can be displayed by the user. Therefore, pop-ups that cannot be displayed by the user should have renderer="false". This will cut down the un-necessary DOM/client components sent to the browser. Similarly the DOM has a panelGroupLayout with a number of cells which are empty. There is no need to send DOM for empty cells.</p>
Do not use hover pop-ups on navigation links.	<p>A hover popup on a navigation link causes the navigation to wait for the hover to be fetched first.</p> <p>Consider removing the hover popup on the compensate workforce table navigation link column and, instead, place it on a separate column or on an icon inside the cell.</p>

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Increase table scrolling timeout.	<p>Tables send a fetch request to the server on a scroll after a timeout. The timeout, before the fetch is sent to the server, is typically only 20ms if the user scrolls a short distance, but can increase to 200ms if the user scrolls further. Therefore performance can be impacted when the user scrolls to the bottom of a page and the table sends multiple requests to the server.</p> <p>To prevent the performance impact, consider increasing the timeout limit to 300ms.</p>
Use a timeout to call <code>_prepareForIncompleteImages</code> .	<p>During Partial Page Rendering (PPR) some images may not load completely. When this occurs, the parent component must be notified that the size of one of its descendants has changed. In the past this was done by using the "complete" attribute on the image tag. Now with Internet Explorer 8 the complete attribute is always false to alleviate performance issues with Internet Explorer 7 and 8. The attribute shows as false even for cached images immediately after the PPR content is fetched.</p> <p>For Internet Explorer 8 use a timeout (10ms) to call <code>_prepareForIncompleteImages</code> so that the image tag called right after the .xml request is processed. Note that this is not an issue for Mozilla Firefox or Google Chrome.</p>
Cache the <code>GetFirstVisibleRowKeyandRow</code> .	<p>Performance can be improved by locally caching the first visible Rowkey and row. This cached value can be deleted on a scroll or a resize.</p>
Use partial page navigation.	<p>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.</p> <p>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.</p> <p>Some known limitations of this feature are:</p> <ul style="list-style-type: none"> ■ 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.

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Use page templates.	<p>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.</p> <p>There are some important considerations when using templates:</p> <ul style="list-style-type: none"> ■ 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 <code><f:subview></code> tag. However, from a performance perspective, this is not typically recommended since it can impact memory usage on the server side. (<code><f:subview></code> 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 <code><f:subview></code> unless it is required. It is not necessary to use <code><f:subview></code> around <code><jsp:include></code> if you can ensure that all IDs are unique. For example, if you are using <code><jsp:include></code>, break a large page into multiple pieces for easier editing. And whenever possible, avoid using <code><f:subview></code>. If you are including content developed by someone else, use <code><f:subview></code> if you do not know which IDs the developer used. In addition, you do not have to put <code><f:subview></code> 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 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Enable ADF rich client geometry management.	<p>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:</p> <ul style="list-style-type: none"> PanelAccordion PanelStretchLayout PanelTabbed BreadCrumbs NavigationPane PanelSplitter Toolbar Toolbox Table Train <p>Notes:</p> <ul style="list-style-type: none"> ■ 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.	<p>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.</p> <p>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.</p>

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Use ADF Rich Client Partial Page Rendering (PPR).	<p data-bbox="656 260 1458 422">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.</p> <p data-bbox="656 432 1458 594">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.</p> <p data-bbox="656 604 1458 661">Two main Ajax patterns are implemented with partial page rendering (PPR):</p> <ul data-bbox="656 672 974 741" style="list-style-type: none"> <li data-bbox="656 672 974 703">■ native component refresh <li data-bbox="656 709 974 741">■ cross-component refresh <p data-bbox="656 751 1458 808">While the framework builds in native component refresh, cross-component refresh has to be done by developers in certain cases.</p> <p data-bbox="656 819 1458 1108">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 user 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 <i>Developing Fusion Web Applications with Oracle Application Development Framework</i>.</p> <p data-bbox="656 1119 1458 1335">Consider a typical situation in which a page includes an <code>af:inputText</code> component, an <code>af:commandButton</code> component, and an <code>af:outputText</code> component. When the user enters a value for the <code>af:inputText</code>, then clicks the <code>af:commandButton</code>, the input value is reflected in the <code>af:outputText</code>. Without PPR, clicking the <code>af:commandButton</code> 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 <code>af:outputText</code> component. To achieve this, you would do two things:</p> <ul data-bbox="656 1346 1458 1602" style="list-style-type: none"> <li data-bbox="656 1346 1458 1430">■ Set up the <code>af:commandButton</code> for partial submit by setting the <code>partialSubmit</code> attribute to <code>true</code>. Doing this causes the command component to start firing partial page requests each time it is clicked. <li data-bbox="656 1440 1458 1602">■ Define which components are to be refreshed when the partial submit takes place, in this example the <code>af:outputText</code> component, by setting the <code>partialTriggers</code> attribute for each of them to the id of the component triggering the refresh. In this example, this means setting the <code>partialTriggers</code> attribute of the <code>af:outputText</code> component to give the id of the <code>af:commandButton</code> component. <p data-bbox="656 1612 1458 1669">The steps above achieve PPR using a command button to trigger the partial page refresh.</p> <p data-bbox="656 1680 1458 1814">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.</p> <p data-bbox="656 1824 1458 1929">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.</p>

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Use ADF rich client navigation.	<p>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.</p> <p>There is one drawback in this approach, however. 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.</p>
Cache resources.	<p>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 additional round trips to the server. To avoid this, make sure all the resources are present on the server.</p> <p>Consider using the ResourceServlet to configure web.xml to enable resource caching:</p> <pre data-bbox="578 800 1105 1024"> <servlet-mapping> <servlet-name>resources</servlet-name> <url-pattern>/js/*</url-pattern> </servlet-mapping> <servlet-mapping> <servlet-name>resources</servlet-name> <url-pattern>/images/*</url-pattern> </servlet-mapping> </pre>
Reduce the size of state token cache.	<p>This property is defined in web.xml org.apache.myfaces.trinidad.CLIENT_STATE_MAX_TOKENS in "token"-based client-side state saving and determines how many tokens should be preserved at any one time. The default value is 15. When this value is exceeded, state will be "forgotten" for the least recently viewed pages, which can impact users that actively use the Back button or that have multiple windows open simultaneously.</p> <p>In order to reduce live memory per session, consider reducing this value to 2. Reducing the state token cache to 2 means one Back button click is supported. For applications without support for a Back button, this value should be set to 1.</p>

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Define custom styles at the top of the page.	<p>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.</p> <p>An HTML page basically has two parts, the "head" and the "body". When you put an <code>af:document</code> component on your page, this component creates both parts of the page for you. Any child component of the <code>af:document</code> 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.</p> <p>To get a component (or static CDATA content) to display in the "head", use the "metaContainer" facet as follows:</p> <pre data-bbox="656 646 1360 1339"> <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> </pre> <p>By default, JDeveloper places the <code>af:document</code> component at the top of the consuming page. However, in some use cases, you may want to place this component in a page fragment or template. If you want to implement <code>af:document</code> through a template, you should find and remove duplicate <code>af:document</code> tags outside of the template definition.</p> <p>If you include <code>af:document</code> and <code>af:form</code> in the template definition, you can expose anything that you may want to customize in those tags through the page template attributes and page template <code>af:facetRef</code>. Your templates are then able to utilize the <code>metaContainer</code> 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.</p> <p>See the <i>Developing Fusion Web Applications with Oracle Application Development Framework</i> for details about <code>af:facetRef</code>.</p>

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Optimize custom JavaScript code.	<p>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: <code><af:resource type="javascript" source=" " /></code>.</p> <p>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.</p>
Disable debug output mode.	<p>The debug-output element in the <code>trinidad-config.xml</code> file specifies whether output should be more verbose to help with debugging. When set to <code>TRUE</code>, 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.</p> <p>Set the debug-output element to <code>FALSE</code>, or if necessary, remove it completely from the <code>trinidad-config.xml</code> file.</p>
Disable test automation.	<p>Enabling test automation parameter <code>oracle.adf.view.rich.automation.ENABLED</code> generates a client component for every component on the page which can negatively impact performance.</p> <p>Set the <code>oracle.adf.view.rich.automation.ENABLED</code> parameter value to <code>FALSE</code> (the default value) in the <code>web.xml</code> file to improve performance.</p>
Disable animation.	<p>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 <code>trinidad-config.xml</code></p>
Disable client-side assertions.	<p>Assertions on client-side code base can have a significant impact on client-side performance. Set the parameter value to <code>FALSE</code> (the default value) to disable client-side assertions. Also ensure that the <code>oracle.adf.view.rich.ASSERT_ENABLED</code> is not explicitly set to <code>TRUE</code> in the <code>web.xml</code> file.</p>
Disable JavaScript Profiler.	<p>When the JavaScript <code>oracle.adf.view.rich.profiler.ENABLED</code> profiler is enabled, an extra round-trip occurs on every page in order to fetch the profiler data. Disable the profiler in the <code>web.xml</code> file to avoid this extra round-trip.</p>
Disable resource debug mode.	<p>When resource debug mode is enabled, the response headers do not tell the browser that resources (JS libraries, CSS style sheets, or images) can be cached.</p> <p>Disable the <code>org.apache.myfaces.trinidad.resource.DEBUG</code> parameter in the <code>web.xml</code> file to ensure that caching is enabled.</p>
Disable timestamp checking.	<p>The <code>org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION</code> parameter controls whether jsp or jsp files are checked for modifications each time they are accessed.</p> <p>Ensure that the parameter value <code>org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION</code> is set to <code>FALSE</code> (the default value) in the <code>web.xml</code> file.</p>

Table 9–2 (Cont.) Configuration Parameters for ADF Faces

Configuration Recommendation	Description
Disable checking for CSS file modifications.	The <code>org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION</code> parameter controls when CSS file modification checks are made. To aid in performance, this configuration option defaults to <code>false</code> - do not check for css file modifications. Set this to <code>TRUE</code> 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 <code>DISABLE_CONTENT_COMPRESSION</code> parameter from the <code>web.xml</code> file or set it to <code>FALSE</code> . For debugging, turn off the style class content compression. You can do this by setting the <code>DISABLE_CONTENT_COMPRESSION</code> property to <code>TRUE</code> .
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 <code>org.apache.myfaces.trinidad.DEBUG_JAVASCRIPT</code> parameter to <code>FALSE</code> in the <code>web.xml</code> file. There are two ways to check that the code is obfuscated using Firefox with Firebug enabled: Check the download size: <ol style="list-style-type: none"> 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: <ol style="list-style-type: none"> 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 11g 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 <code>oracle.adf.view.rich.libraryPartitioning.DISABLED</code> property is set to <code>false</code> in the <code>web.xml</code> file.

9.2.3 Tuning ADF Faces Component Attributes

Table 9–3 provides configuration recommendations for ADF Faces Component Attributes:

Table 9–3 ADF Faces Component Attributes

Configuration Recommendation	Description
Use the "immediate" attribute.	<p>ADF Rich Client components have an <code>immediate</code> attribute. If a component has its <code>immediate</code> attribute set to <code>TRUE</code> (<code>immediate="true"</code>), then the validation, conversion, and events associated with the component are processed during the <code>applyRequestValues</code> phase. These are some cases where setting <code>immediate</code> to <code>TRUE</code> can lead to better performance.</p> <ul style="list-style-type: none"> ■ The <code>commandNavigationItem</code> in the <code>navigationPane</code> can use the <code>immediate</code> attribute set to <code>TRUE</code> 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, <code>immediate</code> should be set to <code>TRUE</code>. In case of an error it be detected earlier in the cycle and additional processing be avoided. <p>ADF Rich Client is built on top of JSF and uses standard JSF lifecycle. See "Understanding the JSF and ADF Faces Lifecycles" in <i>Developing Web User Interfaces with Oracle ADF Faces</i>.</p> <p>There are some important issues associated with the <code>immediate</code> attribute. Refer to "Using the Immediate Attribute" in <i>Developing Web User Interfaces with Oracle ADF Faces</i> for more information.</p> <p>Note that this is an advanced feature. Most of the performance improvements can be achieved using the <code>af:subform</code> component. Refer to <i>Developing Web User Interfaces with Oracle ADF Faces</i> for <code>af:subform</code> details.</p>
Use the "visible" and "rendered" attributes.	<p>All ADF Faces Rich Client display components have two properties that dictate how the component is displayed on the page:</p> <ul style="list-style-type: none"> ■ The <code>visible</code> property specifies simply whether the component is to be displayed on the page, or is to be hidden. ■ The <code>rendered</code> property specifies whether the component shall exist in the client page at all. <p>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.</p>
Use client-side events.	<p>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:</p> <ul style="list-style-type: none"> ■ 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.

Table 9–3 (Cont.) ADF Faces Component Attributes

Configuration Recommendation	Description
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.	<p>ADF Rich Client framework has client-side components that play a role in client-side event handling and component behavior. The <code>clientComponent</code> attribute is used to configure when (or if) a client-side component should be generated. Setting <code>clientComponent</code> attribute to <code>TRUE</code> has a performance impact, so determine if its necessary to generate client-side components.</p> <p>For more information, see "What Happens When You Set <code>clientComponent</code> to true" in <i>Developing Web User Interfaces with Oracle ADF Faces</i>.</p>
Set the <code>childCreation</code> attribute on <code>af:popup</code> to <code>deferred</code> for a server-side performance enhancement	<p>Setting <code>childCreation</code> to <code>deferred</code> postpones construction of the components under the popup until the content is delivered. A deferred setting can therefore reduce the footprint of server-side state in some cases.</p> <p>CAUTION: This approach CANNOT be used if any of the following tags are present inside the popup:</p> <ul style="list-style-type: none"> ■ <code>f:attribute</code> ■ <code>af:setPropertyListener</code> ■ <code>af:clientListener</code> ■ <code>af:serverListener</code> <p>It also CANNOT be used if you need to refer to any child components of the popup before the popup is displayed. Setting <code>childCreation="deferred"</code> will postpone creating any child components of the popup and you cannot refer to them until after the popup is shown.</p>

9.2.4 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 9–4](#) provides some performance recommendations.

Table 9–4 Table and Tree Component Configurations

Configuration Recommendation	Description
Use editingMode="clickToEdit".	<p>When using editingMode="editAll" all content of the editable values holders and their client components is sent. This can significantly increase the payload and the Document Object Model (DOM) content on the client.</p> <p>Consider switching to editingMode="clickToEdit" to reduce the amount of transmitted data and potentially improve user interaction.</p>
Reduce fetchSize when possible.	<p>A larger fetch size attribute on af:table implies that more data needs to be processed, fetched from the server, and displayed on the client. This can also increase the amount of DOM displayed on the client.</p>
Modify table fetch size.	<p>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.</p> <p>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 <code>{bindings.<name>.rangeSize}</code> and should be equal to the iterator size.</p> <p>For more information see "Using Tables and Trees" in <i>Developing Web User Interfaces with Oracle ADF Faces</i>.</p>
Disable column stretching.	<p>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.</p>
Consider using header rows and frozen columns only when necessary.	<p>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.</p>
Consider using visitTree instead of invokeOnComponent.	<p>A partial visit using visitTree is always at least as fast as invokeOnComponent. In addition, for components controlling visiting, providing both invokeOnComponent and visitTree implementations is a source of errors. Consider deprecating invokeOnComponent and use visitTree instead.</p> <p>For more information see "Using Tables and Trees" in <i>Developing Web User Interfaces with Oracle ADF Faces</i>.</p>

9.2.5 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.

9.2.6 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)	<p>Lazy delivery should be used for tables, or other stamped components, which are known to have slow fetch time. The examples are stamped components are the ones based on data controls using web services calls or other data controls with slow data fetch. Lazy delivery can also 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 will be shorter, and the user perceives better performance.</p> <p>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.</p> <p>Executing data fetches in parallel also speeds up the total time to fetch data. This gives an advantage to lazy loading in cases of multiple, and possibly slow, data fetches. While streaming is the default mechanisms to deliver data in lazy mode, parallel execution of data controls is not. In order to enable parallel execution, open the page definition and change <code>RenderHint</code> on the iterator to <i>background</i>.</p> <p>In certain situations, the advantage of parallel execution is faster response time. Parallel execution could potentially use more resources due to multiple threads executing request in parallel and possibly more database connections will be opened.</p> <p>Consider using parallel execution only when there are multiple slow components on the page and the stamped components belong to different data control frames (such as isolated taskflows). Since parallel execution synchronizes on the data control frame level, when there is a single data control frame parallel execution may not improve performance.</p>
Immediate Delivery	<p>Immediate delivery (<code>contentDelivery="immediate"</code>) 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.</p> <p>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.</p>

9.2.7 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 9–5](#) provides some configuration recommendations for DVT components:

Table 9–5 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.	<p>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:</p> <pre>writing-mode:tb-rl;filter:flipV flipH;</pre> <p>While vertical text can look better in some cases, it has a performance impact when the Firefox browser is used.</p> <p>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.</p> <p>To have the best performance, consider using horizontal text instead of vertical text.</p>

9.3 Advanced Tuning Considerations

After you have performed the tuning modifications recommended in the previous section, you can make additional changes that are specific to your ADF Server deployment. Consider carefully whether the recommendations in this section are appropriate for your environment.

9.3.1 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.

Note: When using ADFm, consider using deferred execution and monitor the refresh conditions to maintain performance.

9.3.1.1 Session Timeout Tuning

For ADF applications with a significant user community, the amount of memory held by sessions waiting to expire can negatively impact performance when the default session timeout of 45 minutes is used. The memory being held can be higher than what is physically available, causing the server to not be able to handle the load. For large numbers of users, such as those using a public facing website, the session timeout should be as short as possible.

To improve performance, consider modifying the default session timeout value (in minutes) in the `web.xml` file. Use a session timeout value that works with your use case scenario. The example below shows a session timeout of 10 minutes:

```
<session-config>
  <session-timeout>
```

```

10
</session-timeout>
</session-config>

```

9.3.1.2 View Objects Tuning

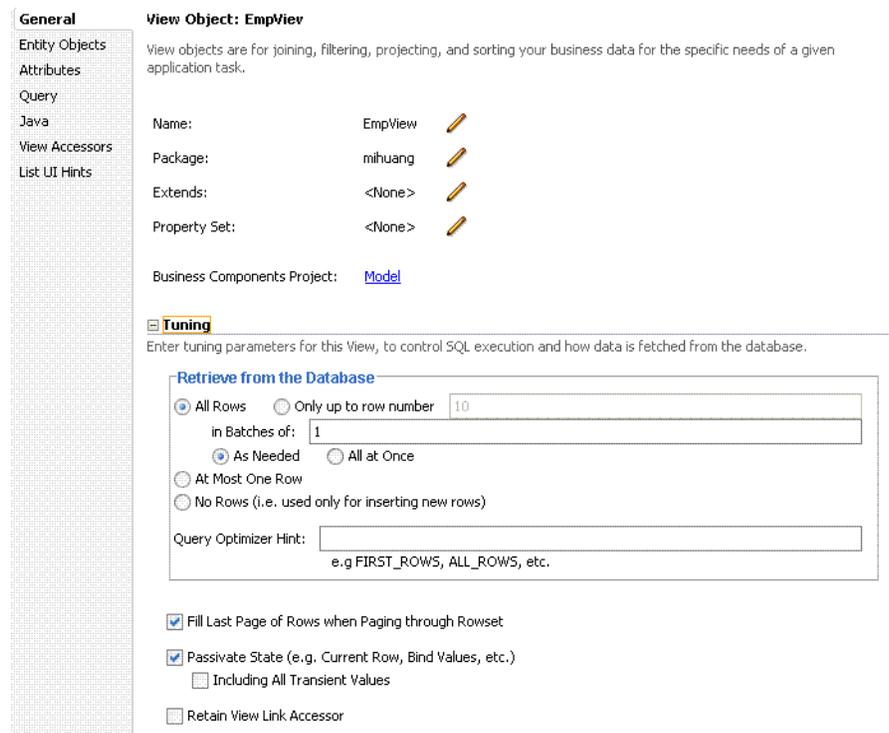
View objects (VOs) provide many tuning options to enable a developer to tailor the View Object to the application's specific needs. View Objects should be configured to use the minimal feature set required to fulfill the functional requirement. The *Developing Fusion Web Applications with Oracle Application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

9.3.1.2.1 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	<p>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:</p> <ul style="list-style-type: none"> ■ Non-updatable EO-based View Objects ■ Expert-mode View Objects <p>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.</p> <p>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 <code>ReadOnly="true"</code> on the <code>EntityUsage</code> attribute in the View Object XML definition.</p>
Insert-only View Objects	<p>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 <code>No Rows</code> in the <code>Retrieve from the Database</code> group box in the View Objects Overview tab. This sets <code>MaxFetchSize</code> to 0 (zero) for the View Object definition.</p>
run time-created View Objects	<p>View Objects can be created at run time using the <code>createViewObjectFromQueryStmt()</code> API on the AM. However, avoid using run time-created View Objects unless absolutely necessary due to potential performance impacts and complexity of tuning.</p>

9.3.1.2.2 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 9–1](#).

Figure 9–1 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 a 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 <code>setForwardOnly(true)</code> on the View Object. Setting forward-only can also prevent caching previous sets of rows as the data set is traversed.

9.3.1.2.3 Additional View Object Configurations Table 9–6 provides additional tuning considerations when using the View Object:

Table 9–6 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 <code>setRangeSize(N)</code> followed by <code>setAccessMode(ResultSet.RANGE_PAGING)</code> 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 <code>scrollToRangePage(P)</code> 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 <code>jbo.use.pers.coll=true</code> . 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 <code>Oracle</code> . To manually override the SQL style, pass the parameter <code>-Djbo.SQLBuilder="SQL92"</code> 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 Entity Object 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.

9.3.1.3 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 Entity Object (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"`).

9.3.1.4 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.

9.3.1.5 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 note 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.
- Use lazy loading for application modules. In the Application Module tuning section, customize runtime instantiation behavior to use lazy loading. Lazy loading can also be set JVM-wide by adding the following JVM argument:

```
-Djbo.load.components.lazily=true
```

9.3.1.6 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.

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 (`-Dproperty=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 9–7 Application Module (AM) Pool Tuning

Configuration Recommendation	Description
Optimize the number of AM pools in the application.	<p>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.</p> <p>For example, 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).</p> <p>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.</p>
Optimize the number of database connections.	<p>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.</p> <p>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.</p>

9.3.1.6.1 General AM Pool Configurations The following guidelines can be used as a general starting point when tuning AM and AM pool behavior. More specific tuning for memory or CPU usage can be found in [Section 9.3.1.6.2, "Configuring AM Pool Sizing"](#).

Table 9–8 AM Pool Tuning Parameters

Parameter	Description
Initial Pool Size <code>jbo.ampool.initpoolsize</code>	<p>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.</p> <p>Configure this value to 10% more than the anticipated number of concurrent AM instances required to service all users.</p>
Maximum Pool Size <code>jbo.ampool.maxpoolsize</code>	<p>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.</p>
Minimum Available Size <code>jbo.ampool.minavailablesize</code>	<p>The minimum number of available application module instances that the pool monitor should leave in the pool during a resource cleanup operation, when the server is under light load.</p> <p>Set to 0 (zero) if you want the pool to shrink to contain no instances when all instances have been idle for longer than the idle time-out after a resource cleanup.</p> <p>The default is 5 instances.</p> <p>While application module pool tuning allows different values for the <code>jbo.ampool.minavailablesize</code> <code>jbo.ampool.maxavailablesize</code> parameters, in most cases it is fine to set these minimum and maximum tuning properties to the same value.</p>

Table 9–8 (Cont.) AM Pool Tuning Parameters

Parameter	Description
Maximum Available Size <code>jbo.ampool.maxavailablesize</code>	<p>The ideal maximum number of available application module instances in the pool when the server is under load.</p> <p>When the pool monitor wakes up to do resource cleanup, it will try to remove available application module instances to bring the total number of available instances down to this ideal maximum. Instances that have been not been used for a period longer than the idle instance time-out will always get cleaned up at this time, then additional available instances will be removed if necessary to bring the number of available instances down to this size.</p> <p>The default maximum available size is 25 instances. Configure this to leave the maximum number of available instances desired after a resource cleanup. A lower value generally results in more application module instances being removed from the pool on a cleanup.</p> <p>While application module pool tuning allows different values for the <code>jbo.ampool.maxavailablesize</code> <code>jbo.ampool.minavailablesize</code> parameters, in most cases it is fine to set these minimum and maximum tuning properties to the same value.</p>
Referenced Pool Size <code>jbo.recyclethreshold</code>	<p>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).</p> <p>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.</p> <p>Configure this value 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.</p>
Maximum Instance Time to Live <code>jbo.ampool.timetolive</code>	<p>The number of milliseconds after which to consider an connection instance in the pool as a candidate for removal during the next resource cleanup regardless of whether it would bring the number of instances in the pool below <code>minavailablesize</code>.</p> <p>The default is 3600000 milliseconds of total time to live (which is 3600 seconds, or one hour). A lower value reduces the time an application module instance can exist before it must be removed at the next resource cleanup. The default value is sufficient for most applications. A higher value increases the time an application module instance can exist before it must be removed at the next cleanup.</p>
Idle Instance Timeout <code>jbo.ampool.maxinactiveage</code>	<p>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.</p> <p>The default is 600000 milliseconds of idle time (which is 600 seconds, or ten minutes). A lower value results in more application module instances being marked as a candidate for removal at the next resource cleanup. A higher value results in fewer application module instances being marked as a candidate for removal at the next resource cleanup.</p>

Table 9–8 (Cont.) AM Pool Tuning Parameters

Parameter	Description
Pool Polling Interval	The length of time in milliseconds between pool resource cleanup.
<code>jbo.ampool.monitorsleepinterval</code>	<p>While the number of application module instances in the pool will never exceed the maximum pool size, available instances which are candidates for getting removed from the pool do not get "cleaned up" until the next time the application module pool monitor wakes up to do its job.</p> <p>The default is to have the application module pool monitor wake up every 600000 milliseconds (which is 600 seconds, or ten minutes). Configuring a lower interval results in inactive application module instances being removed more frequently to save memory. Configuring a higher interval results in less frequent resource cleanups.</p>
Failover	Specifies whether to disable or enable failover. By default, failover is disabled. To enable failover, set the parameter to <code>true</code> .
<code>jbo.dofailover</code>	<p>NOTE: When enabling application module state passivation, a failure can occur when Oracle WebLogic Server is configured to forcibly release connection back into the pool. A failure of this type produces a <code>SQLException</code> (Connection has already been closed) that is saved to the server log. The exception is not reported through the user interface.</p> <p>To ensure that state passivation occurs and changes are saved, set an appropriate value for the <code>weblogic-application.xml</code> deployment descriptor parameter <code>inactive-connection-timeout-seconds</code> on the <code><connection-check-params> pool-params</code> element.</p> <p>Setting the deployment descriptor parameter to several minutes, in most cases, should avoid forcing the inactive connection timeout and the resulting passivation failure. Adjust the setting as needed for your environment.</p>
Locking Mode	Specifies the locking mode (<code>optimistic</code> or <code>pessimistic</code>). The default is <code>pessimistic</code> , which means that a pending transaction state can be created on the database with row-level locks. With <code>pessimistic</code> locking mode, each time an AM is recycled, a rollback is issued in the JDBC connection. Web applications should set the locking mode to <code>optimistic</code> to avoid creating the row-level locks.
<code>jbo.locking.mode</code>	
Database Connection Pooling	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 <code>false</code> , which means that an AM instance can retain its database connection when the AM instance is returned to the AM pool. When set to <code>true</code> , 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.
<code>jbo.doconnectionpooling</code>	
Transaction Disconnect Level	When used in conjunction with <code>jbo.doconnectionpooling=true</code> , specifies BC4J behavior for maintaining JDBC ResultSets. By default <code>jbo.txn.disconnect_level</code> is 0, and passivation can be used to close any open ResultSets when the database connection is disconnected from the AM instance. Configuring <code>jbo.txn.disconnect_level</code> to 1 can prevent this behavior to avoid the passivation costs for this situation.
<code>jbo.txn.disconnect_level</code>	

For parameters that can be configured for memory-constrained systems, see [Table 9–9](#).

Table 9–9 AM Pool Sizing Configurations - Memory Considerations

Parameter	Description
Initial Pool Size 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.
Maximum Pool Size 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.
Minimum Available Pool Size 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.
Maximum Available Pool Size 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 9–10](#).

Table 9–10 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.

9.3.1.6.2 Configuring AM Pool Sizing The Application Module pool sizing configuration is largely dependant on the number of concurrent users you expect to have. To prevent performance issues, you need to make sure AM pool size is sufficient to serve all concurrent users.

Caution: The following example assumes at least 100 concurrent users. Always consult your own use case scenarios to determine the appropriate settings for your deployment.

To configure these parameters, open the `setDomainEnv.sh` file for the WebLogic Server instance and find these lines:

```
JAVA_OPTIONS="${JAVA_OPTIONS}"
export JAVA_OPTIONS
```

Replace these lines with the following:

```
JAVA_OPTIONS="-Djbo.ampool.doampooling=true
-Djbo.ampool.minavailablesize=1
-Djbo.ampool.maxavailablesize=120
-Djbo.recyclethreshold=60
-Djbo.ampool.timetolive=-1
-Djbo.load.components.lazily=true
-Djbo.doconnectionpooling=true
-Djbo.txn.disconnect_level=1
-Djbo.connectfailover=false
```

```
-Djbo.max.cursors=5
-Doracle.jdbc.implicitStatementCacheSize=5
-Doracle.jdbc.maxCachedBufferSize=19 ${JAVA_OPTIONS}"
```

Note: To limit performance implications, set the `ampool.maxavailablesize` to a value that is at least 20% more than the maximum number of concurrent users you expect in your own use case scenarios.

9.3.1.6.3 AM Pool Resource Cleanup Configurations These parameters affect the frequency and characteristics for AM pool resource cleanups.

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 9–11](#) for more information.

Table 9–11 AM Pool Resource Cleanup Configurations - Memory Considerations

Parameter	Description
<code>jbo.ampool.minavailablesize</code>	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
<code>jbo.ampool.maxavailablesize</code>	A lower value generally results in more AM instances being removed from the pool on a cleanup.
<code>jbo.ampool.timetolive</code>	A lower value reduces the time an AM instance can exist before it must be removed at the next resource cleanup.
<code>jbo.ampool.maxinactiveage</code>	A low value results in more AM instances being marked as a candidate for removal at the next resource cleanup.
<code>jbo.ampool.monitorsleepinterval</code>	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 9–12 AM Pool Resource Cleanup Configurations - CPU Considerations

Parameter	Description
<code>jbo.ampool.minavailablesize</code> and <code>jbo.ampool.maxavailablesize</code>	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.
<code>jbo.ampool.timetolive</code>	A higher value increases the time an AM instance can exist before it must be removed at the next resource cleanup.
<code>jbo.ampool.maxinactiveage</code>	A higher value results in fewer AM instances being marked as a candidate for removal at the next resource cleanup.
<code>jbo.ampool.monitorsleepinterval</code>	Configuring a higher interval results in less frequent resource cleanups.

9.3.1.7 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.

9.3.1.8 Defer Task Flow Execution

By default, task flows are activated when the page is loaded, even when the task flow is not initially rendered. This causes unnecessary overhead if the task flow is never displayed.

Note: For regions and taskflows, the amount of time it takes to evaluate the current viewId and the time it takes to calculate input parameters to the flow can impact your overall performance. Consider this during your design phase.

9.3.1.9 Task Flow in a Popup

By default, the child components under a popup are created even when popup is not accessed. To avoid this overhead, consider the following:

- Set childCreation to deferred
Set childCreation="deferred" on the popup
Set activation="deferred" on the taskflow

Caution: This approach cannot be used if any of the following tags are present inside the popup:

- f:attribute
- af:setPropertyListener
- af:clientListener
- af:serverListener

t also cannot be used if you need to refer to any child components of the popup before the popup is displayed. Setting childCreation="deferred" will postpone creating any child components of the popup and you cannot refer to them until after the popup is shown. In that case, use Conditional Activation as described below:

- Use Conditional Activation
Add property listener on the popup in the jsff to set a condition
Set activation="conditional" on the taskflow
Set activate=<condition> on the taskflow

9.3.1.10 Configuring the Task Flow Inside Switcher

By default, task flows under switchers are activated when the page is loaded, not when the switcher facet is displayed. To avoid this, use conditional activation and set "active" to an expression language (EL) expression that returns 'true' when the facet is displayed.

9.3.1.11 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 *Developing Fusion Web Applications with Oracle Application Development Framework*.

9.3.1.12 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 *Developing Fusion Web Applications with Oracle Application Development Framework*.

Tuning Oracle TopLink

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 10.1, "About Oracle TopLink and EclipseLink"](#)
- [Section 10.2, "Basic Tuning Considerations"](#)
- [Section 10.3, "Advanced Tuning Considerations"](#)

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

- "Understanding Queries" at <http://www.eclipse.org/eclipselink/documentation/2.5/concepts/queries.htm#CHDGGCJB>
- "Understanding Caching" at <http://www.eclipse.org/eclipselink/documentation/2.5/concepts/general004.htm#CHDEEBFG>
- "Understanding Mappings" at <http://www.eclipse.org/eclipselink/documentation/2.5/concepts/mappingintro.htm#CHDFEJIJ>

For more information on Oracle TopLink, see the **TopLink** page on the Oracle Technology Network (OTN).]

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.

10.2 Basic Tuning Considerations

The following tuning recommendations are applicable to most deployments. Always consult your own usecase scenarios before implementing any of these configurations.

- [Creating Efficient SQL Statements and Queries](#)
- [Tuning Cache Configuration](#)
- [Tuning the Mapping and Descriptor Configurations](#)
- [Using Data Partitioning](#)

10.2.1 Creating Efficient SQL Statements and Queries

This section covers using efficient SQL statements and SQL querying. [Table 10–1](#) and [Table 10–2](#) show tuning parameters and performance recommendations related to SQL statements and querying.

Table 10–1 EJB/JPA Using Efficient SQL Statements and Querying

Tuning Parameter	Description	Performance Notes
Parameterized SQL Binding	<p>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.</p> <p>See Also: "Understanding Caching" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDEFHHEH and "Understanding Querying" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/queries.htm#CHDGGCJB</p> <p>Default Value: PERSISTENCE_UNIT_DEFAULT (which is true by default)</p>	<p>Leave parameterized SQL binding enabled for selected databases and JDBC drivers that support these options.</p>
JDBC Statement Caching	<p>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.</p> <p>Note: For Java EE applications, use the data source's statement caching (and do not use EclipseLink Statement Caching for EJB3.0/JPA, for example: <code>eclipselink.jdbc.cache-statements="true"</code>).</p> <p>Set this option in an Oracle Weblogic data-source by setting Statement Cached Type and Statement Cached Size configuration options.</p> <p>See also "Increasing Performance with the Statement Cache" in <i>Administering JDBC Data Sources for Oracle WebLogic Server</i>.</p> <p>Default Value: The Oracle Weblogic Server data source default statement cache size is 10 statements per connection.</p>	<p>You should always enable statement caching if your JDBC driver supports this option. The Oracle JDBC driver supports this option.</p>

Table 10–1 (Cont.) EJB/JPA Using Efficient SQL Statements and Querying

Tuning Parameter	Description	Performance Notes
Fetch Size	<p>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.</p> <p>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.</p> <p>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.</p> <p>Note: The default value means use the JDBC driver default value, which is typically 10 rows for the Oracle JDBC driver.</p> <p>To configure this, use query hint "eclipselink.jdbc.fetch-size".</p> <p>Default Value: 0</p>	<p>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.</p>
Batch Writing	<p>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.</p> <p>The persistence property in persistence.xml "eclipselink.jdbc.batch-writing"="JDBC" is used to configure this.</p> <p>Default Value: Off</p>	<p>Enable for the persistence unit.</p>
Change Tracking	<p>This is an optimization feature that lets you tune the way EclipseLink detects changes in an Entity.</p> <p>Default Value: AttributeLevel if using weaving (Java EE default), otherwise Deferred.</p>	<p>Leave at default AttributeLevel for best performance.</p>
Weaving	<p>Can disable through persistence.xml properties "eclipselink.weaving"</p> <p>Default Value: On</p>	<p>Leave on for best performance.</p>

Table 10–1 (Cont.) EJB/JPA Using Efficient SQL Statements and Querying

Tuning Parameter	Description	Performance Notes
Read Only	<p>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.</p> <p>Set through query hint "eclipselink.read-only".</p> <p>Can also be set at entity level using @ReadOnly class annotation.</p> <p>Default Value: False</p>	For optimal performance use read-only on any query where the resulting objects are not changed.
firstResult and maxRows	<p>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.</p>	Use on queries that can have a large result set and only a subset of the objects is needed.
Sequence number pre-allocation	<p>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.</p> <p>Default Value: 50</p>	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.

10.2.1.1 Tuning Entity Relationships Query Parameters

Table 10–2 shows the Entity relationship query parameters for performance tuning.

Table 10–2 EJB3.0 Entity Relationship Query Performance Options

Tuning Parameter	Description	Performance Notes
Batch Fetching	<p>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.</p> <p>Batch fetching has three types: JOIN, EXISTS and IN. The type is set through the query hint "eclipselink.batch.type"</p> <p>Note that batching is only allowed on queries that have a single object in their select clause. The query hint to configure this is "eclipselink.batch". Batch fetching can also be set using the @BatchFetch annotation.</p> <p>Default Value: Off</p>	<p>Use for queries of tables with columns mappings to table data you need. You should only use either batch fetching 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.</p> <p>Batch fetching is more efficient than joining because it avoids reading duplicate data; therefore for best performance for queries where batch fetching is supported, consider using batch fetching instead of join reading.</p>
Join Fetching	<p>Join fetching 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.</p> <p>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.</p> <p>You can specify the use of join in JPQL (<code>JOIN FETCH</code>), or you can set it multi-level in a query hint, "eclipselink.join-fetch". It also can be set in the mapping annotation @JoinFetch.</p> <p>Joining is part of the JPA specification, whereas batch fetching is not. And, joining works on queries that not work with batch fetching. 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 fetching does not.</p> <p>See Also: "Join Fetching" at http://www.eclipse.org/eclipselink/documentation/2.5/solutions/performance001.htm#CHDEGCHH</p> <p>Default Value: Not Used</p>	<p>Use for queries of tables with columns mappings to table data you need. You should only use either batch fetching 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 fetching is not supported, a join is recommended</p>

Table 10–2 (Cont.) EJB3.0 Entity Relationship Query Performance Options

Tuning Parameter	Description	Performance Notes
Lazy loading	<p>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.</p> <p>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.</p> <p>See Also: "Using Lazy Loading" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/mappingintro001.htm#CEGBCJAG</p> <p>Default Value: On for collection mapping (ToMany mappings, @OneToMany, @ManyToMany)</p> <p>Default Value: Off for reference (ToOne mappings, @OneToOne, @ManyToOne)</p> <p>(Note that setting lazy loading On for @OneToOne, @ManyToOne requires weaving, which is On by default for Java EE.)</p>	<p>Use lazy loading for all mappings. Using lazy loading and querying the referenced objects using batch fetching or Join is more efficient than Eager loading.</p> <p>You may also consider using optimized loading with LoadGroups which allows a query to force instantiation of relationships.</p>

10.2.2 Tuning Cache Configuration

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 10.3.1, "Integrating with Oracle Coherence"](#).

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 10–3](#) shows the cache configuration options.

For more information on cache configuration, see "Understanding Caching" at <http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDFHHEH>.

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 10–3](#).

Table 10–3 EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Object Cache	<p>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.</p> <p>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.</p> <p>The annotation type <code>@Cacheable</code> specifies whether an entity should be cached. Caching is enabled when the value of the persistence.xml caching element is <code>ENABLE_SELECTIVE</code> or <code>DISABLE_SELECTIVE</code>. The value of the <code>Cacheable</code> annotation is inherited by subclasses; it can be overridden by specifying <code>Cacheable</code> on a subclass.</p> <p><code>Cacheable(false)</code> means that the entity and its state must not be cached by the provider.</p> <p>Default Value: Enabled (shared is True)</p>	<p>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. Also, consider disabling the cache for infrequently accessed entities</p>
Query Result Set Cache	<p>In addition to the object cache in EclipseLink, EclipseLink also supports a query cache:</p> <ul style="list-style-type: none"> ■ 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. <p>The query hints for a query cache are:</p> <p>"eclipselink.query-cache"</p> <p>"eclipselink.query-cache.size"</p> <p>"eclipselink.query-cache.invalidation"</p> <p>See Also: "Understanding Caching" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDEFHHEH and "JPA Query Customization Extensions" at http://www.eclipse.org/eclipselink/documentation/2.5/jpa/extensions/queryhints.htm#sthref498</p> <p>Default Value: Not Used</p>	<p>Use for frequently executed non-primary key queries with infrequently changing result sets. Use with a cache invalidation time out to refresh as needed.</p>

Table 10–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Cache Size	<p>Cache size can be configured through persistence properties: "eclipselink.cache.size.<entity>" "eclipselink.cache.size.default" "eclipselink.cache.type.default"</p> <p>See Also: "About the Persistence Unit" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/appdeployment002.htm#BABHCJDG and "Class PersistenceUnitProperties" at http://www.eclipse.org/eclipselink/api/2.3/org/eclipse/persistence/config/PersistenceUnitProperties.html</p> <p>Default Value: Type <code>SoftWeak</code>, Size 100 (per Entity). The default value may be different if <code>TopLink</code> is running on Exalogic. See "Enable the Exalogic Automated Tuner" in the <i>Solutions Guide for Oracle TopLink</i> for more information about the Exalogic default.</p>	<p>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.</p> <p>Consider creating larger cache sizes for entities that have many instances that are frequently accessed and stale data is not a big issue.</p> <p>Consider using smaller cache sizes or no cache for frequently updated entities that must always have fresh data, or infrequently accessed entities.</p>
Locking	<p>Oracle supports the locking policies shown in Table 10–4: no locking, optimistic, pessimistic, and read-only.</p> <p>Locking is set through JPA <code>@Version</code> annotation, <code>eclipselink.read-only</code></p> <p>See "Descriptors and Locking" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/descriptors002.htm#CHEEEIEA</p> <p>Default Value: No Locking</p>	<p>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.</p>

Table 10–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Cache Usage	<p>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.</p> <p>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 <code>CheckCacheByExactPrimaryKey</code>.</p> <p>Default Value: <code>DoNotCheckCache</code></p>	<p>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 <code>CheckCacheByExactPrimaryKey</code>).</p> <p>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).</p>
Isolation	<p>There is not a single tuning parameter that sets a particular database transaction isolation level in a JPA application that uses EclipseLink.</p> <p>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:</p> <ul style="list-style-type: none"> ■ Locking mode ■ Use of the Session Cache ■ External Applications ■ Database Login method <code>setTransactionIsolation</code> <p>See Also: "Isolated Cache" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache001.htm#CDEEGICF</p>	

Table 10–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options

Tuning Parameter	Description	Performance Notes
Cache Refreshing	<p>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.</p> <p>Refreshing can be enabled at the entity level (<code>alwaysRefresh</code> or <code>refreshOnlyIfNewer</code> and <code>expiry</code>) and at the query level (with the <code>eclipselink.refresh</code> query hint). You can also force queries to go to the database with (<code>disableHits</code>). Using an appropriate locking policy is the only way to ensure that stale or conflicting data does not get committed to the database.</p> <p>For more information see: Section 10.2.2.1, "Cache Refreshing Scenarios"</p> <p>See Also: "Understanding Caching" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDEFHHEH</p> <p>Default Value: No Cache Refreshing</p>	<p>Try to avoid entity level cache refresh and instead, consider configuring the following:</p> <ul style="list-style-type: none"> ■ cache refresh on a query-by-query basis ■ cache expiration ■ isolated caching

10.2.2.1 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.

10.2.2.2 Tuning the Locking Mode Policies

The locking modes, as shown in [Table 10–4](#), 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 "Descriptors and Locking" at

<http://www.eclipse.org/eclipselink/documentation/2.5/concepts/descriptors02.htm#CHEEEIEA>.

Table 10–4 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. 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: "Using Optimistic Locking" at http://www.eclipse.org/eclipselink/documentation/2.5/concepts/mappingintro005.htm#CEGDIIB	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.	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".	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

10.2.3 Tuning the 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, "Mapping and Descriptors" at

<http://www.eclipse.org/eclipselink/documentation/2.5/solutions/performance002.htm#sthref153>.

10.2.4 Using Data Partitioning

EclipseLink allows you to configure data partitioning using the `@Partitioned` annotation. Partitioning enables an application to scale information across multiple databases; including clustered databases. For more information on using `@Partitioned` and other partitioning policy annotations, see "Partitioning Annotations" at http://www.eclipse.org/eclipselink/documentation/2.5/jpa/extensions/annotations_ref.htm#CACHIHB.

10.3 Advanced Tuning Considerations

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.

- [Integrating with Oracle Coherence](#)
- [Analyzing EclipseLink JPA Entity Performance](#)

10.3.1 Integrating with Oracle Coherence

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 "Grid Cache Configuration" in *Integrating Oracle Coherence*.

For more information on Oracle Toplink integration with Oracle Coherence, see "Introduction to Toplink Grid with Oracle Coherence" in *Integrating Oracle Coherence*.

10.3.2 Analyzing EclipseLink JPA Entity Performance

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

- For form monitoring performance, see "Performance Monitoring" at <http://www.eclipse.org/eclipselink/documentation/2.5/concepts/monitoring003.htm#BABJABIH>. Note that this tool is intended to profile and monitor information in a multithreaded server environment.
- For profiling performance, see "Task 1: Measure EclipseLink Performance with the EclipseLink Profiler" at <http://www.eclipse.org/eclipselink/documentation/2.5/solutions/performance002.htm#CHDIAFJI>. 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 best performance, remember to restore the logging levels to the default levels when you are done profiling or debugging.

Part IV

SOA Suite Components

This part covers how to tune Oracle SOA Suite components to improve performance'

The SOA Suite components are documented in the following chapters:

- [Chapter 11, "Tuning the SOA Infrastructure"](#)
- [Chapter 12, "Tuning Oracle BPEL Process Manager"](#)
- [Chapter 13, "Tuning Oracle Mediator"](#)
- [Chapter 14, "Tuning Oracle Managed File Transfer"](#)
- [Chapter 15, "Tuning Oracle Business Rules"](#)
- [Chapter 16, "Tuning Oracle Business Process Management"](#)
- [Chapter 17, "Tuning Oracle Human Workflow"](#)
- [Chapter 18, "Tuning Oracle Business Activity Monitoring"](#)
- [Chapter 19, "Tuning Oracle Service Bus"](#)
- [Chapter 20, "Tuning Oracle Enterprise Scheduler Service"](#)

Tuning information for B2B, Healthcare Integration, and adapters are documented elsewhere. You can find how to tune for performance using the links provided.

- [User's Guide for Oracle B2B](#)
- [Healthcare Integration User's Guide for Oracle SOA Suite](#)
- ["Oracle JCA Adapter Tuning Guide" in *Understanding Technology Adapters*](#)

Tuning the SOA Infrastructure

This chapter describes how to tune the properties of the SOA Infrastructure.

- [Section 11.1, "About the SOA Infrastructure"](#)
- [Section 11.2, "Tuning SOA Work Managers"](#)
- [Section 11.3, "Tuning SOA Infrastructure Parameters"](#)
- [Section 11.4, "Using Advanced Tuning Options"](#)
- [Section 11.5, "Advanced Tuning for Work Managers"](#)

11.1 About the SOA Infrastructure

The SOA Infrastructure is a Java EE-compliant application running in Oracle WebLogic Server. The application manages composites and their lifecycle, service engines, and binding components. See "Introduction to the SOA Infrastructure Application" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for more information.

This chapter describes some concepts and procedures you can use to tune SOA Infrastructure components using Work Managers and other tuning parameters.

This chapter does not cover any diagnostic tools or methodologies that are needed for a holistic approach but talks about isolated tuning options for isolated symptoms. For information on monitoring the SOA Infrastructure performance to pinpoint problem areas, see "Monitoring the SOA Infrastructure" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

11.2 Tuning SOA Work Managers

This section lists a few simple checks and configurations you can make to take advantage of Work Managers.

Starting in Oracle SOA Suite 12c (12.1.3), Work Managers handle most SOA-related work threads. For more details on how Work Managers manage threads and self-tune, see "Understanding Work Managers" in *Administering Server Environments for Oracle WebLogic Server*.

Before attempting to configure Work Managers, you should have a good understanding of your environment and be able to quantify the following:

- Volume of incoming requests that you need processed
- Internal processing requirements including any SLA expectations for transactions

- An understanding of the processes you have that do not use Work Managers, such as the Event Delivery Network and most adapters

Based on the information collected above, you can take advantage of the Work Managers' self-tuning feature by accomplishing the following tasks:

- [Configuring database connections with the SOADataSource property](#)
- [Configuring Work Managers with the SOAMaxThreadsConfig attribute](#)

11.2.1 Configuring database connections with the SOADataSource property

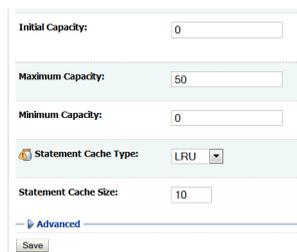
The SOADataSource property determines the total number of concurrent database connections that are available for your SOA processes. Because SOA processes use the database for most of their activities, this is a very important setting and can create a bottleneck if not appropriately configured.

In order to tune this setting, it is important to understand your database resources and consult your DBA.

To tune SOADataSource, do the following:

1. Log in to the Oracle WebLogic Server Administration Console.
2. Select **Services** from the left-hand menu and then choose **DataSources**.
3. On the **DataSource** configuration page, select **SOADataSource**.
4. Select the **Connection Pool** tab and scroll down to find the **Maximum Capacity** attribute.

The default for the **Maximum Capacity** attribute is 50. For most practical use cases, you should set this value to 300 to increase the size of the entire SOADataSource connection pool.



Initial Capacity:	<input type="text" value="0"/>
Maximum Capacity:	<input type="text" value="50"/>
Minimum Capacity:	<input type="text" value="0"/>
Statement Cache Type:	<input type="text" value="LRU"/>
Statement Cache Size:	<input type="text" value="10"/>

[Advanced](#)

The SOADataSource setting is leveraged by the SOAMaxThreadConfig configuration explained in [Section 11.2.2](#). SOADataSource defines the total number of connections available to all Work Managers, while the SOAMaxThreadConfig attribute defines what percentage of those connections will be available to certain categories of Work Managers.

11.2.2 Configuring Work Managers with the `SOAMaxThreadsConfig` attribute

SOA composites are associated with a group of Work Managers that handles various components and functional areas. The `SOAMaxThreadsConfig` attribute determines the number of threads allowed for different groups of SOA Work Managers in a domain.

The number of threads allotted to handle incoming requests, internal processes, and other SOA processes are defined as percentages of the `SOADataSource` property explained in [Section 11.2.1](#). The default percentage values and categories of the `SOAMaxThreadsConfig` attribute are listed in [Table 11–1](#).

Table 11–1 Thread distributions for Work Managers determined by `SOAMaxThreadsConfig`

Group	Description
<code>incomingRequestsPercentage</code> Default: 20%	This parameter determines the percentage of threads that your system allocates to Work Managers that process incoming client requests.
<code>internalBufferPercentage</code> Default: 30%	This parameter determines the percentage of threads distributed to other SOA functions, such as EDN and adapters.
<code>internalProcessingPercentage</code> Default: 50%	This parameter determines the percentage of threads that your system allocates to Work Managers for internal processes.

This attribute is defined at the domain level and will apply to all the Work Managers under that domain. You can set this attribute using the `SoaInfraConfig` MBean in the Fusion Middleware Control MBean Browser.

To access the attribute:

1. Log in to Fusion Middleware Control.
2. Select System MBean Browser from the WebLogic Domain menu.
3. In the System MBean Browser folder structure, navigate through the following folders: **Application Defined MBeans** --> **oracle.as.soainfra.config** --> **Server: *AdminServerName*** --> **SoaInfraConfig** --> **soa-infra**
4. When you click on **soa-infra**, its attributes will be listed in the main pane on the right. Look for the `SOAMaxThreadsConfig` attribute and click on it. You should then see the parameters and values listed in [Table 11–1](#).

Click **Apply** when you are ready to make your changes.

Remember that the values you are adjusting on this screen are percentages, not the discrete number of threads. You should ascertain the total number of threads available to you by checking the value of the `SOADataSource` property, which is described in [Section 11.2.1](#).

In a sample scenario where `SOADataSource` is set to 50 connections and you kept the default `SOAMaxThreadConfig` percentages listed in [Table 11–1](#), you would have the following thread allocations:

- 20% of 50 = 10 threads to process incoming request
- 30% of 50 = 15 threads for processes not using work managers
- 50% of 50 = 25 threads to process internal processes

11.3 Tuning SOA Infrastructure Parameters

[Table 11–2](#) describes the optimal settings for parameters with the greatest impact on SOA Infrastructure performance.

Table 11–2 Essential SOA Infrastructure Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
<p>AuditLevel</p> <p>Default: Production</p>	<ul style="list-style-type: none"> ■ High database CPU ■ Contentions causing increased processing times in applications 	<p>To prevent possible performance degradation, maintain the lowest audit level possible or retain the default of Production. Avoid setting the audit level to Development whenever possible.</p> <p>This parameter can be set in the Enterprise Manager. You can find the Audit Level parameter page on the SOA Infrastructure Common Properties page.</p> <p>To find this page:</p> <ol style="list-style-type: none"> 1. Toggle the SOA folder in your left-hand Target Navigation. 2. Right-click on the <code>soa-infra (soa_server)</code> you want to tune. 3. Select SOA Administration --> Common Properties <p>For more information about this parameter, see "Configuring Oracle SOA Suite and Oracle BPM Suite Profiles" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i>.</p>	<p>Decreasing the audit level will cause the system to generate less audit data.</p> <p>Diagnosing performance issues and general troubleshooting may be more difficult.</p>
<p>Audit Purge Policy</p> <p>Default: Everyday Midnight and purges records older than 7 days</p>	<ul style="list-style-type: none"> ■ Exponential growth in database size ■ If configured at peak hours, purging can take resources from other processes 	<ul style="list-style-type: none"> ■ Ensure auto purge is enabled ■ Perform purges more often ■ Set the auto purge to kick off at a time when there is less resource contention from other processes <p>For information on finding the Auto Purge page in the Oracle Enterprise Manager Fusion Middleware Control, see "Deleting Large Numbers of Instances with Oracle Enterprise Manager Fusion Middleware Control" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i>.</p>	<p>Disabling this feature will make maintaining on-going database growth more time-consuming.</p>

11.4 Using Advanced Tuning Options

The rest of this chapter provides additional performance tuning settings for SOA that you might want to consider. These options are not listed in any specific order. Before changing any of these properties, you should have a holistic knowledge of your environment, SOA processes, and non-SOA processes.

It is important to understand that any advanced performance optimization should be a customized approach for individual scenarios, settings, environments, and expectations. A customized approach requires detailed capturing of diagnostic information in order to pinpoint and isolate bottlenecks and areas that need optimization.

For information on monitoring the SOA Infrastructure performance to pinpoint problem areas, see "Monitoring the SOA Infrastructure" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

This section covers the following topics:

- [Enabling the Composite Lazy Loading Feature](#)
- [Changing Modularity Profiles](#)
- [Tuning Your Database for SOA Processes](#)
- [Tuning Event Delivery Network Parameters](#)
- [Tuning the WebLogic Server](#)

11.4.1 Enabling the Composite Lazy Loading Feature

Composite lazy loading is a new feature in 12c. It improves server startup time when there is a large number of composites deployed.

At server startup, composites are loaded minimally, meaning that they will only create in-memory java models and MBeans. Any initializing tasks, such as loading components and resources used by composite, viz. WSDL and Schema file, are loaded later at first-request time when they are needed.

This greatly improves server startup times and staggers the composite startup times for when they receive requests, reducing overhead from rarely used or retired composites.

Composite lazy loading is helpful for:

- Scenarios requiring speedy disaster recovery times during a server failure
- Customers with a huge number of composites that use large WSDLs or schema files

Composite lazy loading is enabled by default and can be configured at the domain level and at the composite levels. See the following sections for more details:

- [Configuring Composite Lazy Loading for the Domain level](#)
- [Configuring Composite Lazy Loading at the Component level](#)

11.4.1.1 Configuring Composite Lazy Loading for the Domain level

Composite lazy loading is enabled by default at the domain level. This setting can be disabled from System MBean Browser in Enterprise Manager for Fusion Middleware Control. Changes to this setting will take affect when the server restarts.

To change the setting for lazy loading feature for the domain level:

1. After logging into Enterprise Manager, right-click on the domain you want to tune from the list of the WebLogic domains in the **Target Navigation** browser.
2. Select **System MBean Browser** from the drop-down menu.
3. In the System MBean Browser folder structure, navigate through the following folders: **Application Defined MBeans** --> **oracle.as.soainfra.config** --> **Server: AdminServerName** --> **SoaInfraConfig** --> **soa-infra**
4. When you click on **soa-infra**, its attributes will be listed in the main pane on the right. Look for the `CompositeLazyLoading` attribute and click on it.
5. On the `CompositeLazyLoading` page, you can set the value to `true` to enable it or `false` to disable it. Click **Apply** when you are ready to make your changes.

11.4.1.2 Configuring Composite Lazy Loading at the Component level

By default, composites will inherit the lazy loading setting from the domain level. If there is a use case where you would like to control this behavior at specific composite level, then this can be configured in the `composite.xml`, which is a file generated when you create a new SOA Suite composite application.

You can find `composite.xml` in the home folder of the application you want to edit. You can also edit `composite.xml` by accessing it in JDeveloper. For more information on `composite.xml`, see "What Happens When You Create a SOA Application and Project" in *Developing SOA Applications with Oracle SOA Suite*.

At the beginning of the `composite.xml` of the application that you want to edit, you will need to add the new property `lazyLoading="false"` to override the default behavior at the domain level. Then redeploy the composite.

Below is a sample code snippet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by Oracle SOA Modeler version 12.1.3.0.0 at [8/7/13 4:14 PM]. -->
<composite name="ValidatePayment"
  revision="1.0"
  label="2013-08-07_16-14-11_843"
  mode="active"
  state="on"
  lazyLoading="false"
  xmlns="http://xmlns.oracle.com/sca/1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  .....
  .....
</composite>
```

11.4.2 Changing Modularity Profiles

Modularity is another 12c feature that helps improve your memory footprint and server startup times. Some profile options are limited to only components and features that are used by your selected composites. The modularity profile you select determines what components will be loaded in memory.

12c has out-of-box profiles that can be changed after completing installation. By default, new 12c customers will have `SOA_FOUNDATION` as their install profile. Existing customers upgrading to 12c will have `SOA_CLASSIC` as their install profile by default.

[Table 11-3](#) shows the modularity profiles in the increasing order of memory footprint size.

Table 11–3 Modularity Profiles

Profile	Components
BPEL-ONLY	BPEL Components + SOA Common Infrastructure + Partial Adapter set
ORCHESTRATION	BPEL-Only + HWF + Partial Adapter set
SOA FOUNDATION	Orchestration + Mediator + Rules + Partial Adapter set
Default for new 12c customers	
SOA FOUNDATION ENTERPRISE	SOA Foundation + Full Adapter Set
SOA FOUNDATION WITH B2B	SOA Foundation Enterprise + B2B
SOA FOUNDATION WITH HEALTHCARE	SOA Foundation with B2B + Healthcare UI
SOA CLASSIC	SOA Foundation with B2B + BPM Modules
Default for upgrade customers	

If you are using a limited set of components or features in the SOA suite, you can change your profile to optimize your memory usage and server startup times. This can free up resources for crucial processes and can improve disaster recovery.

You can change your modularity profile from the SOA dashboard in Enterprise Manager for Fusion Middleware Control.

For more information on how to change your profile, see "Configuring SOA Infrastructure Properties" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* to find the **SOA Infrastructure Common Properties** page.

Then see "Configuring Oracle SOA Suite and Oracle BPM Suite Profiles" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for more information on the profiles.

11.4.3 Tuning Your Database for SOA Processes

This section describes advanced strategies for tuning your database for SOA processes. Make sure you have already read and followed the general database tuning suggestions covered in [Section 2.6](#) of this book before progressing.

11.4.3.1 Collecting Optimizer Statistics

Optimizer statistics provide details about the database and the objects in the database. The query optimizer uses these statistics to choose the best execution plan for each SQL statement. See "Introduction to the Query Optimizer" in the *Oracle Database SQL Tuning Guide* for more information.

11.4.3.1.1 Automatic Statistics Gathering Because objects in a database can change constantly, you must update statistics regularly so that they accurately describe these objects.

All SOA databases should use the Automatic Statistics Collection, which is enabled by default. This job runs every night. See "Controlling Automatic Optimizer Statistics Collection" in the *Oracle Database SQL Tuning Guide* for more information.

11.4.3.1.2 Manual Statistics Gathering Automatic optimizer statistics collection is sufficient for most database objects, but in a database that is close to going live or for tables that are modified/purged significantly, manual statistic gathering is needed. See

"Gathering Optimizer Statistics Manually" in *Oracle Database SQL Tuning Guide* for more information.

For SOA databases that implement purging of stale data on regular basis, you should collect stats manually right after purging has completed. In these cases, use the `DBMS_STATS.GATHER_TABLE_STATS` procedure. See "DBMS_STATS" in *Oracle Database PL/SQL Packages and Types Reference* for how to do this.

11.4.3.1.3 Optimizing the MDS Database Repository With Statistics Ensure that automatic statistics collection is enabled. See "Controlling Automatic Optimizer Statistics Collection" in the *Oracle Database SQL Tuning Guide* for more information.

In most cases, the first 32 characters of `PATH_FULLNAME` in the `MDS_PATHS` table are the same. You can prevent the database from putting them in the same section of the histogram by doing the following:

1. Drop the histogram for `PATH_FULLNAME` column by executing a command structured like the following as system:

```
execute dbms_stats.delete_column_stats(ownname=>'mdsSchemaOwner',
tabname=>'MDS_PATHS', colname=>'PATH_FULLNAME', col_stat_type=> 'HISTOGRAM');
```

2. Set table preferences to exclude collecting histogram for the `PATH_FULLNAME` column with a command structured like the following:

```
execute dbms_stats.set_table_prefs(mdsSchemaOwner, 'MDS_PATHS', 'METHOD_OPT',
'FOR COLUMNS SIZE 1 PATH_FULLNAME');
```

11.4.3.2 Tuning Temporary Tablespaces for SOA

See [Section 2.6.2](#) for general guidelines on tuning TEMP tablespaces for Oracle Fusion Middleware before progressing to this section.

Some SOA queries can generate a large amount of disk sorts that require high amounts of temporary space. Therefore, the use of multiple temporary tablespaces and tablespace groups is recommended to meet these requirements and assure optimal performance.

The suggested minimum size for the TEMP tablespace or tablespace group assigned to the SOA schema owner is 6 GB with auto-extend enabled. See "Changing Data File Size" in the *Oracle Database Administrator's Guide* for more information on how to resize a tablespace and enable auto-extend.

11.4.3.3 Minimizing SOA Database Contention

Most SOA workloads generate heavy DML activity in the database and are likely to experience contention on database objects.

Wait event data in Automatic Workload Repository (AWR) reports reveal various symptoms that might be impacting performance. The most common wait events that could occur in SOA database are as follows:

- DB CPU
- Db file sequential read, db file scattered read
- log file sync
- enq: HW - contention
- enq: TX - index contention
- buffer busy waits
- gc buffer busy acquire, gc buffer busy release (RAC)

- [enq: SQ - contention](#)

The rest of this section describes how to tune the following wait events:

- [log file sync](#)
- [enq:HW contention](#)
- [enq: TX - index contention](#)

Recommendations for the wait events "gc buffer busy acquire" and "gc buffer busy release" are also discussed under the [enq: TX - index contention](#) subsection.

For information on using AWR reports to identify contentions in your database, see "Comparing Database Performance Over Time" in the *Database Performance Tuning Guide*.

11.4.3.3.1 log file sync In a SOA database, it is very common to see the foreground wait event "log file sync" with a high average wait time. This is caused by redo log performance. The possible reasons for high log file sync waits are as following:

- The database log writer (LGWR) is unable to complete writes fast enough for one of the following reasons:
 - Disk I/O performance to log files is not good enough.
 - LGWR is starving for CPU resources.
- LGWR is unable to post the processes fast enough due to excessive commits.
- LGWR is suffering from other database contentions, such as enqueue waits or latch contention.

Tuning the redo log performance can improve performance for applications running in an Oracle Fusion Middleware environment.

See [Section 2.6.2](#) for general guidelines on tuning redo logs for Oracle Fusion Middleware before using the strategies in this section to tune for SOA processes.

- [Finding LGWR wait events](#)
- [Sizing online redo logs to control the frequency of log switches and minimize system waits](#)
- [Optimizing the redo log disk to prevent bottlenecks](#)
- [Determining the optimal sizing of the log_buffer](#)
- [Tuning the LGWR Process](#)
- [Using Smart Flash Logging for ExaData](#)

Finding LGWR wait events

The first step in identifying the root cause is to find and break down LGWR wait events. You can query for LGWR wait events using its SID, as shown in the following example:

```
SQL> SELECT sid, event, time_waited, time_waited_micro
       FROM v$session_event
       WHERE sid IN
          (SELECT SID FROM v$session WHERE type!='USER' AND program LIKE '%LGWR%' )
       ORDER BY time_waited;
```

Sizing online redo logs to control the frequency of log switches and minimize system waits

The suggested minimum setting for redo logs is to have at least 3 log groups of 2 GB each. Monitor the redo log performance periodically. Then adjust the number of redo log groups and size of each member as appropriate in order to control the frequency of log switches and minimize system waits.

Size the redo log files according to the amount of redos the system generates. A rough guide is to switch logs at most once every 20 minutes.

For example, if your online redo logs switches once every 5 minutes during peak database activity, the logs would each need to be 4 times larger than their current size to achieve the 20 minute guideline. The calculation for this is $20\text{min} / 5\text{min} = 4x$.

Optimizing the redo log disk to prevent bottlenecks

A SOA database is highly write-intensive, which generates massive amount of redo per second and per transaction. Sometimes no amount of disk tuning may relieve redo log bottlenecks, because Oracle must push all updates for all disks into a single redo location.

If I/O bandwidth is an issue, doing anything other than improving I/O bandwidth is not useful. One way to relieve redo bottlenecks is to use faster redo storage. It is recommended to use Solid State Disk (SSD) redo log files. SSD has greater bandwidth than platter disk.

Determining the optimal sizing of the log_buffer

SOA applications insert, modify, and delete large volumes of data. Most of these operations are committed in a row-by-row fashion rather than in batch mode. Frequent commits cause a significant overhead on the redo performance, so sizing the log_buffer optimally is important for performance.

The statistic REDO BUFFER ALLOCATION RETRIES from your AWR reports and/or from V\$ views reflects the number of times a user process waits for space in the redo log buffer. You can obtain this statistic through the dynamic performance view V\$SYSSTAT with the following query:

```
SELECT NAME, VALUE
FROM V$SYSSTAT
WHERE NAME = 'redo buffer allocation retries';
```

The value of redo buffer allocation retries should be near zero over an interval. If this value increments consistently, then processes have had to wait for space in the redo log buffer. The wait can be caused by the log buffer being too small or by check pointing. You can improve this wait by attempting the following:

- Increase the size of the redo log buffer, if necessary, by changing the value of the initialization parameter LOG_BUFFER. The value of this parameter is expressed in bytes. A good starting rule of thumb for a write intensive workload is to configure the log buffer to 100mb. Use caution while increasing log_buffer setting, because excessive redo size can also cause high "log file sync" waits.
- Improve the check pointing or archiving process.

You can also check to see if the log buffer space wait event is a significant factor in the wait time for the instance. If not, the log buffer size is most likely adequate.

Tuning the LGWR Process

For most SOA workloads, the commit rate is very high, and decreasing commits is not an option. If previous strategies to address high log file sync did not improve redo log performance, try increasing the priority of LGWR or increasing the priority class of LGWR to RT from the command line.

Using Smart Flash Logging for ExaData

If your database is on ExaData machine, it should have a minimum of Bundle Patch 11 (BP11) installed in order to take advantage of the Smart Flash Logging feature.

Exadata Smart Flash Logging is an additional feature implemented in Exadata Storage software 11.2.2.4.2 and database version 11.2.0.2 + BP11. With this feature, 512MB of flash storage is reserved for redo writes and the LGRW process adopts a different pattern of behavior.

In a system which does not use this feature, LGWR writes in parallel to multiplexed copies of the redo logs and then waits for all writes to complete. This means that the time taken to perform these writes (indicated by the Oracle wait interface statistics log file parallel write) is the time taken for the slowest disk to complete the write.

With Exadata Smart Flash Logging, the redo log files remain on disk, but the additional reserved 512MB of space is created on flash storage. When issuing a write call, LGWR writes to the redo logs on disk as usual but will also make a parallel write to the flash area. LGWR then waits for whichever of these writes completes first to post it, after which it continues without waiting for the other.

11.4.3.3.2 enq:HW contention The High Water enqueue contention (enq:HW - contention) occurs when competing processes are inserting into the same table and are trying to increase the high water mark of a table simultaneously.

In a SOA database, this issue is experienced by tables that have large object (LOB) columns, such as CUBE_SCOPE, XML_DOCUMENT, AUDIT_DETAILS, etc. Under a heavy load, LOB segments in these tables experience contention, which is seen in an AWR report as the wait event enq: HW contention.

The default storage for LOBs in an Oracle database is BasicFiles. Frequently allocating extents or reclaiming chunks may cause contention for the LOB segment high water marks. This contention can also occur for LOB segments that are ASSM-managed, since space allocation only acquires one block at a time.

This contention can be eliminated by switching LOB storage from BasicFiles to SecureFiles. SecureFiles is a LOB storage architecture that provides performance benefits over traditional BasicFiles. See "About LOB Storage" in the *Database SecureFiles and Large Objects Developer's Guide* for more information on these two architectures.

Migrating BasicFiles to SecureFiles can be done using one of the following methods:

- Set the database parameter `SECURE_FILES = ALWAYS`.

This method is applicable for new installations prior to creating SOA tables using RCU. Once this parameter is set at the instance level, any new LOB segments created will use SecureFiles automatically.

- Use the online redefinition method.

This method is applicable for installations that already have SOA tables created in them. In such cases, LOB segments from tables in a SOA database experiencing enq: HW contention can be migrated to SecureFiles.

Using the online redefinition method to migrate to SecureFiles can be done with very little downtime. See "Using DBMS_REDEFINITION" in *Oracle Database*

PL/SQL Packages and Types Reference for how to use the `lob_storage_as` parameter to reassign a LOB store.

- Set the database event value to 44951 using the following script:

```
ALTER SYSTEM SET EVENT='44951 TRACE NAME CONTEXT FOREVER, LEVEL 1024?
scope=spfile;
```

This method will help a SOA installation using an Oracle version older than 11g to avoid `enq:HW` contentions on LOB segments.

You can use your AWR and Automatic Database Diagnostic Monitor (ADDM) reports to identify which LOB objects are suffering from `enq:HW` - contentions. For most systems, however, it is highly recommended to move the LOB columns listed in the following table to `SecureFiles`.

Table Name	Column Name	Recommended LOB Storage Attributes
ATTACHMENT	ATTACHMENT	COMPRESS CACHE
AUDIT_DETAILS	BIN	COMPRESS CACHE
CUBE_SCOPE	SCOPE_BIN	COMPRESS CACHE

11.4.3.3 enq: TX - index contention In most SOA scenarios, multiple database sessions will insert thousands of rows into SOA tables. In these situations, the number of index keys is constantly increasing, particularly the primary key indexes.

Though the number of primary key indexes increases over time, B-tree structure indexes only target a few database blocks for key insertions. These B-tree index insertions can become problematic in a Real Application Cluster (RAC). This issue is seen in an AWR report as high "Buffer Busy waits."

B-tree indexes create other contentions for RAC environments that show in an AWR as "gc buffer busy acquire" and "gc buffer busy release" wait events. These occur when a transaction inserting a row in an index has to wait for the end of a different transaction's index block split, forcing the session to wait as well. When many concurrent inserts lead to excessive index block splits, performance decreases.

The solution for these contentions is to create global, hash partitioned indexes. This forces a random distribution of index keys across many database blocks to avoid these contentions or hot spots.

Hash partitioning has proven to be the best tuning method to address index contention. You should use your AWR and ADDM reports to identify indexes that need to be partitioned. Once you have identified hot indexes, consider hash partitioning them to reduce or avoid index contention.

For information on hash partitioning, see "Creating Hash Partitioned Tables and Global Indexes" in the *Database VLDB and Partitioning Guide*.

11.4.3.4 Purging

The need for aggressive and continuous purging is a key aspect to improving performance and controlling disk space in SOA.

Managing auto purge feature, enabled by default to help manage on-going database growth in 12c, is described in [Table 11-2](#). SOA installations that accumulate a lot of data should also implement a purging strategy to clean up redundant data, to help of SQL query performance, and to save disk space.

To create a purging strategy, see "Developing a Purging and Partitioning Methodology" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

11.4.3.5 Reclaiming Space

SOA installations that implement frequent purging of unwanted data from SOA tables are more likely to experience disk space issues.

This problem occurs even with ASSM and locally managed tablespaces. When automatic purge scripts delete rows from database tables and indexes to release space within the data blocks for reuse, space is not released immediately after rows are deleted. This causes fragmentation, with some space too small for reuse, particularly when the tables contain LOB columns.

To alleviate fragmentation and consolidate disk space, you should manually shrink tables and LOB columns to reclaim space on a routine basis.

Use the Segment Advisor to identify segments that would benefit from online segment shrink. Note that most SOA segments should be candidates for online segment shrink operations after constant purging. See "Using the Segment Advisor" in *Oracle Database Administrator's Guide* for more information on how to use the Segment Advisor.

Once you have identified the database tables and indexes that need shrinking, use the following commands to reclaim space manually:

```
ALTER TABLE CUBE_SCOPE ENABLE ROW MOVEMENT;
ALTER TABLE CUBE_SCOPE SHRINK SPACE;
ALTER TABLE CUBE_SCOPE MODIFY LOB (SCOPE_BIN) (SHRINK SPACE);
ALTER TABLE CUBE_SCOPE DISABLE ROW MOVEMENT;
```

This shrink operation consolidates free space below the high water mark and compacts the segment. Then it moves the high water mark and de-allocates space above the high water mark.

11.4.4 Tuning Event Delivery Network Parameters

The Event Delivery Network (EDN) delivers events published by Oracle Mediator, Oracle BPEL Process Manager, and external publishers such as Oracle Application Development Framework entity objects. See "Introduction to the Event Delivery Network" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for a more detailed description.

[Table 11-4](#) lists parameters that you can find in the Fusion Middleware MBean Browser and tune for improved event delivery.

Table 11–4 Event Delivery Network Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
<p>numberOfPollerThreads</p> <p>Default: -1</p>	<ul style="list-style-type: none"> Out-of-resource issues, e.g. out of memory, system overload, transaction issue, etc. Contention with other SOA threads 	<p>The default value of -1 means that the system will use ThreadsPerSubscriber to determine a poller thread count. This is optimal for most configurations.</p> <p>However, if you have a high number of subscribers, the default setting will try to assign a thread to each subscriber. This will slow your system down. You should define a positive integer to limit the amount of poller threads created for this task.</p> <p>See "Updating the Local numberOfPollerThreads Value at the Service Component Level" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> for how to change this parameter's value in the Fusion Middleware MBean Browser.</p>	<p>If the value is too low for your system, then poller threads can cause event backlogs and long latencies between event publishing and composite instance creation.</p> <p>If the value is too high, then excess poller threads will consume server resources needlessly.</p>
<p>ThreadsPerSubscriber</p> <p>Default: 1 thread</p>	<ul style="list-style-type: none"> Out-of-resource issues, e.g. out of memory, system overload, transaction issue, etc. Contention with other SOA threads 	<p>Typically, the default of 1 thread per subscriber is optimal.</p> <p>Note that numberOfPollerThreads should be adjusted first, since that parameter takes precedence over this value.</p> <p>See "Updating the ThreadsPerSubscriber Attribute in the System MBean Browser" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> for how to change this parameter's value in the Fusion Middleware MBean Browser.</p>	<p>If the value is too low for your system, then poller threads can cause event backlogs and long latencies between event publishing and composite instance creation.</p> <p>If the value is too high, then excess poller threads will consume server resources needlessly.</p>

Table 11–5 lists the parameters you can modify for individual business events in JDeveloper. To modify these attributes, right-click on the event you want to edit to bring up the pop-up menu. From this menu, select **Edit Subscribed Events...** or **Edit Published Events...**, depending on the parameter you are trying to edit.

For descriptions of the subscribed event parameters you can edit, see "How to Subscribe to a Business Event" in *Developing SOA Applications with Oracle SOA Suite*.

Table 11–5 Business Event Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
Consistency for a Subscribed Event Default: oneAndOnlyOne	You are experiencing either one or both problems with business event delivery. <ul style="list-style-type: none"> Unfulfilled delivery guarantee requirements to event subscribers Unnecessary system overhead from global transactions 	Set the level for a selected business event to <code>guaranteed</code> in JDeveloper. A guaranteed delivery is performed in a local transaction with only one trip to the main queue. You can also edit this parameter on the Subscriptions page in the Oracle Enterprise Manager Fusion Middleware Control. See "Viewing Business Event Subscribers" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> for details.	<code>oneAndOnlyOne</code> guarantees delivery by taxing resources. If a guaranteed delivery fails, then there are no local retries and a system failure message is generated. Message duplication could occur in the event that the calling global transaction rolls back and retries since the message delivery is outside of that transaction.
Durability for a Subscribed Event Default: Yes	You are experiencing either one or both problems with business event messages. <ul style="list-style-type: none"> Multiple dropped events Unnecessary retention of messages in the system 	Set the value under the <code>Durable</code> column to <code>No</code> to disable durability for a subscribed event using JDeveloper. This will free the system from having to persist messages to storage.	<code>No</code> will cause the system to drop events if the subscriber is not running when events are published. <code>Yes</code> will retain events in the JMS server and incur overhead.
Persistent Delivery for a Published Event Default: yes	<ul style="list-style-type: none"> Unreliable messaging High overhead 	Set this value to <code>No</code> to disable persistent delivery. This will reduce overhead.	<code>No</code> will cause less reliable messaging following an event publish since there is no persistence. <code>Yes</code> will incur overhead by guarding against a JMS server crash.
Time to Live for a Published Event Default: 0 ms	<ul style="list-style-type: none"> Non-expired and unconsumed messages are occupying system resources and requiring manual cleanup. Messages are deleted before subscribers can read them. 	Specify a positive integer so that expired messages are automatically removed from the system and not consumed by the subscribers. The integer represents milliseconds. The best value depends on your system and can be determined by monitoring metrics. Note that the default value of 0 means that messages will never expire.	If the message expiration duration value is too low, published messages can expire before an intended subscriber can read it. Once it is gone, it cannot be retrieved. If the value is too high, then lingering messages can occupy system resources.

11.4.4.1 Adding JMS Topics with Mapping

By default, all events are mapped to a single WLS topic.

If you have a large backlog of events or are experiencing latency or slowness in event processing due to single or limited JMS topics, you should create additional JMS topics and modify events to JMS mapping so that event types of different performance characteristics may be grouped or managed separately.

However, if you do this, the system will have additional JMS topics and JMS artifacts to manage, and you will have mapping changes to consider.

If you choose to add a new JMPS topic, you should consider the following steps:

- [Choosing a JMS Topic Type](#)
- [Creating JMS Topics](#)
- [Mapping Events to JMS Topics](#)

Choosing a JMS Topic Type

You can create either a WLSJMS topic or an AQJMS topic.

WLSJMS is the default JMS topic type. It does not provide database indexing, LOB streaming, embedded rules engines, and lock management as well as AQJMS.

AQJMS will typically not be faster than WLSJMS, but if your system has high concurrences, AQJMS works well because it is single-threaded. AQJMS can also get constrained by lower and storage nodes in Exalogic.

Creating JMS Topics

You can create a new WLSJMS topic under the SOAJMSModule in the WebLogic Administration Console if you are logged in as an Administrator. See "Create topics in a system module" in the *Oracle WebLogic Server Administration Console Online Help* for details on navigating to the **Create a New JMS System Module Resource** and creating a JMS topic.

You can create an AQJMS topic using the Database Navigator in JDeveloper or SQL Developer as soainfra user by running the following script:

```
define edn_user=your_soainfra_schema_username
define topic=your_custom_aqjms_topic_name, e.g. 'EDN_AQJMS_TOPIC_2'
define topic_table=your_custom_aqjms_topic_table, e.g. 'EDN_AQJMS_TOPIC_TABLE_2'

begin
  DBMS_AQADM.stop_queue(queue_name => '&edn_user..&topic');
  DBMS_AQADM.drop_queue(queue_name => '&edn_user..&topic');
  DBMS_AQADM.drop_queue_table(queue_table => '&edn_user..&topic_table');
end;
/
begin
  dbms_aqadm.create_queue_table(queue_table => '&edn_user..&topic_table',
                               queue_payload_type => 'SYS.AQ$_JMS_MESSAGE',
                               multiple_consumers => true);
  dbms_aqadm.create_queue(queue_name => '&edn_user..&topic',
                          queue_table => '&edn_user..&topic_table',
                          max_retries => 256);
  dbms_aqadm.start_queue(queue_name=> '&edn_user..&topic');
end;
/
commit;
```

You can reference "Create a JMS Queue or Topic" in *Administering JMS Resources for Oracle WebLogic Server* for information about AQ JMS topics.

Mapping Events to JMS Topics

When you have created new JMS topics, you can map business events to specific topics. Note that one event type can be mapped to only one JMS topic, whereas one JMS topic can store multiple event types.

For more information on using the Enterprise Manager for Fusion Middleware Control to map events, see "Mapping Business Events to JMS Topic Destinations on the Business Events Page" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

11.4.5 Tuning the WebLogic Server

The performance of the SOA Infrastructure depends on the WebLogic Server. Though tuning the WebLogic Server is a separate task not thoroughly addressed in this book, you can use [Table 11–6](#) to check the tuning knobs that affect the SOA Infrastructure.

Table 11–6 Essential WebLogic Server Tuning for SOA Infrastructure

Parameter	Tuning Recommendation	Resource
ProductionModeEnabled Default: The mode you set during domain creation.	Production mode maximizes performance. You should enable this if you are not developing applications. You can enable the <code>ProductionModeEnabled</code> MBean in Oracle Fusion Middleware Control.	See "Configure general settings" in <i>Administering Oracle WebLogic Server with Fusion Middleware Control</i> . Changing the domain mode will also change certain security and autodeployment settings. See "Development vs. Production Mode Default Tuning Values" in <i>Tuning Performance of Oracle WebLogic Server</i> for more information on domain modes.
WebLogic Server Logging Levels Default: Notification	To reduce the volume of logging requests, use the lowest acceptable logging level, such as <code>ERROR</code> or <code>WARNING</code> whenever possible. You can set log levels for handlers and loggers in a variety of ways.	See "Using Log Severity Levels" in <i>Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server</i> for these methods.
HTTP Access Logging Default: Enabled	By default, the HTTP subsystem keeps a log of all HTTP transactions in a text file. Turn off HTTP access logging to improve performance. You can disable this property using the Oracle WebLogic Server Administration Console.	See "Enable and configure HTTP logs" in the <i>Oracle WebLogic Server Administration Console Online Help</i> .
JMS Persistence and Persistence Storage Default: Enabled	Ensure that the right persistence level is set for the Java Message Service (JMS) destinations. <ul style="list-style-type: none"> ■ For persistent JMS scenarios, there are two choices: <code>File Store</code> and <code>JDBC Store</code>. Typically, operations on a File Store perform better than JDBC Store. If there are multiple JMS servers involved, create each store on a separate disk to lower I/O contention. ■ For non-persistent JMS scenarios, turn off persistence at the JMS server level by un-checking the <code>Store Enabled</code> flag from the Advanced section of the General tab for the JMS server in the WebLogic Server console. You can also override the persistence mode at the JMS destination level. 	See "Using Custom File Stores and JDBC Stores" in <i>Tuning Performance of Oracle WebLogic Server</i> for more information on creating and managing persistent JMS stores. To decide if you should turn off the JMS server level persistence, see "Limitations of the Persistent Store" in <i>Administering Server Environments for Oracle WebLogic Server</i> .
Connection Backlog Buffering	You can tune the <code>Accept Backlog</code> parameter when dealing with a large number of concurrent clients. The <code>Accept Backlog</code> parameter specifies how many TCP connections can be buffered in a wait queue. You can tune the number of connection requests that a WebLogic Server instance will accept before refusing additional requests.	For more information, see "Tuning Connection Backlog Buffering" in <i>Tuning Performance of Oracle WebLogic Server</i> .

11.5 Advanced Tuning for Work Managers

This section details how Work Managers are mapped to SOA projects and specific components, and how you can use some advanced configuration options to fine-tune Work Manager performance. These options include the following tuning strategies:

- [Configuring Fair Share Request Class for SOA Work Managers](#)
- [Creating a New Work Manager Constraint](#)

When SOA Suite is installed, it creates a set of default Work Managers, global Work Managers, and application Work Managers to manage various areas of the SOA Infrastructure.

High priority composites can be associated with a Work Manager group that has been configured for higher priority. [Table 11–7](#) lists the set of Work Managers that are created when SOA is installed and describes the work area they manage.

Table 11–7 Work Manager Descriptions

Work Manager Name	Responsible Area
SOA_Request_WM	SOA synchronous request clients, such as the following: <ul style="list-style-type: none"> ■ Facade invocation ■ WebService client requests ■ Direct/ADF/Rest requests ■ B2B
SOA_Notification_WM	All SOA notification requests.
<i>WorkManagerName_dspSystem</i>	BPEL-specific system dispatcher messages.
<i>WorkManagerName_dspInvoke</i>	BPEL-specific engine process invocation dispatcher messages
<i>WorkManagerName_dspEngine</i>	BPEL engine process dispatcher messages
<i>WorkManagerName_dspNonBlocking</i>	BPEL engine process non-blocking invocation dispatcher messages
<i>WorkManagerName__Analytics</i>	BPEL analytics
<i>WorkManagerName_MediatorParallelRouting</i>	Mediator parallel routing
<i>WorkManagerName_MediatorErrorHandling</i>	Mediator error handling
<i>WorkManagerName_bpmnSystem</i>	BPM system dispatcher messages
<i>WorkManagerName__bpmnInvoke</i>	BPM engine process invocation dispatcher messages
<i>WorkManagerName__bpmnEngine</i>	BPM process engine dispatcher messages
<i>WorkManagerName__bpmnNonBlocking</i>	BPM process non-blocking invocation dispatcher messages
SOA_DataSourceBound_WM	All SOA backend processing services that access <i>SOADataSource</i> , including Workflow Enterprise JavaBeans (EJBs).
SOA_Default_WM	All SOA services that do not access the <i>SOADataSource</i> connection pool. It also handles Case Management.
SOA_EDN_WM	Event Delivery Network (EDN)
<i>WorkManagerName_Adapter</i>	Adapter framework

The *SOAMaxThreadsConfig* property, discussed in [Section 11.2.2](#), determines the number of connections that will be used by Work Managers to process incoming requests, internal processes, and other processes. This configuration determines the optimal usage for each of these processing categories when the system is functioning at its full potential.

Minimum and Maximum Constraints can also be set on Work Managers to control upper and lower limit of connections for Work Managers. A Fair Share Request class for a Work Manager can be created to determine the relative priority assigned to a Work Manager. The constraints and request class mentioned here are the ones most commonly configured for SOA Work Managers.

All SOA Work Managers are pre-configured with request classes and constraints that make most sense. It is strongly recommended to run with the default settings and make any essential changes after an evaluation period.

For information on all Work Manager constraints and request classes you can create and their default behaviors, refer to "Managing Work Manager Groups" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

11.5.1 Configuring Fair Share Request Class for SOA Work Managers

A Fair Share Request Class allows you specify the relative priority of a given Work Manager. All SOA Work Managers managing internal process have been configured to one of the two Fair Share Classes that are created by default: `soa_fairShare_20` and `soa_fairShare_80`, with fair share values set to 20 and 80 respectively. A Fair Share value is a relative value from 1 to 1000.

If you want to further tune SOA Work Manager priorities, you will need to create new Fair Share classes. For more information on how to do this, see "Viewing and Creating Work Manager Groups" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

11.5.2 Creating a New Work Manager Constraint

In addition to the default categories available in `SOAMaxThreadConfig`, you can create new categories to address specific scenarios.

Some processes in SOA do not require database connections. These processes do not depend on SOA Data Source allocation and hence will not have wait for available connections.

The SOA Infrastructure automatically creates Work Managers that manage most of your processes and allocate resources accordingly. For most cases, performance can be improved by leveraging existing Work Managers and tuning their performance using some of the knobs described above.

If you have special scenarios where you would like to handle uniquely, you can create a new Work Manager and configure it to meet special circumstances. You will be either creating a new application or a web application Work Manager. See "Viewing and Creating Work Manager Groups" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for detailed procedures.

Tuning Oracle BPEL Process Manager

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 12.1, "About BPEL Process Manager"](#)
- [Section 12.2, "Tuning BPEL Parameters"](#)
- [Section 12.3, "Using Other Tuning Strategies"](#)

12.1 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 an overview of Oracle BPEL Process Manager, see "Oracle Business Process Execution Language (BPEL) Process Manager" under "Key Components" in *Understanding Oracle SOA Suite*.

12.2 Tuning BPEL Parameters

This section includes tuning recommendations for BPEL parameters that are *likely* or *highly likely* to improve performance. For descriptions of the other tuning parameters available for SOA Components, see the component-specific chapters in this guide and *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

For detailed information on how to monitor, configure, and manage BPEL process service components and service engines, see the chapters listed in "Administering BPEL Process Service Components and Engines" of *Administering Oracle SOA Suite and Oracle Business Process Management Suite*. Also see "Using the BPEL Process Service Component" in the *Developing SOA Applications with Oracle SOA Suite* for how to use sensors to monitor select BPEL activities.

12.2.1 Tuning BPEL Engine

This section describes the performance tuning properties at the BPEL engine level. They can be configured using the Enterprise Manager Fusion Middleware Control. For information on using Oracle Enterprise Manager Fusion Middleware Control to configure and monitor parameters, see "Getting Started with Administering Oracle

SOA Suite and Oracle BPM Suite" and "Accessing the System MBean Browser from the Component Property Pages" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

12.2.1.1 Tuning BPEL Engine Parameters

[Table 12–1](#) lists the essential tuning parameter you can adjust to improve performance for the BPEL engine.

Table 12–1 Essential BPEL Engine Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
auditLevel Default: Inherit	You are experiencing low performance because of frequent database inserts into the audit_trail table.	Use the Off value to stop storing audit information. Note that the auditLevel is set at the SOA Infrastructure level. See "Configuring BPEL Process Service Engine Properties" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> to see how to find and tune this parameter.	This property sets the audit trail logging level for both durable and transient processes. If you turn this off, both business flow and payload tracking is disabled. You will not be able to view the state of BPEL processes in the Oracle Enterprise Manager Console.

[Table 12–2](#) describes additional BPEL engine parameters that can be tuned for small performance improvements. Note that for most use cases, the default value is the recommended value.

Table 12–2 Other BPEL Engine Tuning Knobs

Parameter	Description
SyncMaxWaitTime Default: 45 seconds.	<p>You can decrease this parameter's value to improve performance.</p> <p>The SyncMaxWaitTime property sets the maximum time the process result receiver waits for a result before returning. This property is required for synchronous interactions and applicable to transient processes.</p> <p>See "How To Specify Transaction Timeout Values" in <i>Developing SOA Applications with Oracle SOA Suite</i> for instructions on how to find this property in the System MBean Browser of Oracle Enterprise Manager Fusion Middleware Control.</p>
largedocumentthreshold Default: 10000 (100 kilobytes)	<p>You can decrease this parameter's value to improve performance.</p> <p>This property sets the maximum size (in kilobytes) of a BPEL variable before it is stored in a separate table from the rest of the instance scope data. It is applicable to both durable and transient processes.</p> <p>Large XML documents can slow performance if they are constantly read in and written out whenever processing on an instance must be performed.</p> <p>See "Configuring BPEL Process Service Engine Properties" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> to see how to find and tune this parameter in the Enterprise Manager Fusion Middleware Control.</p>
validateXML Default: False	<p>You should set this parameter to the default value of False to improve performance.</p> <p>This property can make the Oracle BPEL Process Manager intercept nonschema-compliant payload data by validating incoming and outgoing XML documents. However, XML payload validation can slow performance.</p> <p>You can find this parameter in the System MBean Browser. See "Configuring BPEL Process Service Engine Properties" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> for how to find advanced BPEL properties using the More BPEL Configuration Properties... button from the BPEL Service Engine Properties page in Enterprise Manager Fusion Middleware Control.</p>
InstanceKeyBlockSize Default: 10000 keys	<p>You can increase the instance key block size to a value greater than the number of updates to the ci_id_range table to improve performance.</p> <p>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.</p> <p>See "Configuring BPEL Process Service Engine Properties" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> to see how to find and tune this parameter using the System MBean Browser in Enterprise Manager Fusion Middleware Control.</p>
Audit Level Threshold Default: 10000	<p>You can decrease this parameter's value to improve performance.</p> <p>This property sets the maximum size (in kilobytes) of an audit trail details string before it is stored separately from the audit trail. Strings larger than the threshold setting are stored in the audit_details table instead of the audit_trail table. In cases where the variable is very large, performance can be severely impacted by logging it to the audit trail.</p> <p>See "Configuring BPEL Process Service Engine Properties" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> to see how to find and tune this parameter in Enterprise Manager Fusion Middleware Control.</p>

12.2.2 Tuning BPEL in a Composite

This section lists some BPEL properties that can be tuned for individual composites to improve performance. The BPEL properties set inside a composite affect the behavior of the component containing the BPEL process only. Each BPEL process can be created as a component of a composite.

BPEL composite properties can be modified in `composite.xml` using JDeveloper, or in the System MBean Browser of Oracle Enterprise Manager Fusion Middleware Control. For in-depth descriptions of each property's function, see "Deployment Descriptor Properties" of *Developing SOA Applications with Oracle SOA Suite*.

The BPEL tuning considerations listed in [Table 12–3](#) may not be applicable to all BPEL deployments. Always consult your own use case scenarios to determine if these configurations should be used in your deployment. See "How to Define Deployment Descriptor Properties in the Property Inspector" in *Developing SOA Applications with Oracle SOA Suite* for how to find and edit the parameters listed below.

Table 12–3 Essential BPEL in a Composite Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
OneWayDeliveryPolicy Default: <code>async.persist</code>	Slow performance because resources are being used to persist delivery messages.	Set value to <code>async.cache</code> . Incoming delivery messages for durable processes will be kept only in the in-memory cache. By default, incoming requests are saved in the delivery service database table <code>dlv_message</code> .	This setting has a high risk of losing messages or overloading the system. It will also change the threading model for adapter.
Audit Policy Default: All activities	Slow performance because every activity is being audited.	Audit only key activities.	Lower level activities will not have an audit trail.
<code>inMemoryOptimization</code> Default: <code>False</code>	Slow performance because the <code>completionPersistPolicy</code> parameter has been activated at the BPEL component level, causing the BPEL server to dehydrate either all or some instances.	Set value to <code>False</code> to tell the Oracle BPEL Server that this process is a transient process and dehydration is not required.	No dehydration means that activities in the instance will be lost if the system crashes.

[Table 12–4](#) describes additional BPEL parameters that can be tuned for small performance improvements, but in most cases, the default value is the recommended value. For in-depth descriptions of each property's function, see "Properties for the partnerLinkBinding Deployment Descriptors" of *Developing SOA Applications with Oracle SOA Suite*.

Table 12–4 Other BPEL in a Composite Tuning Knobs

Parameter	Description
idempotent Default: True	An idempotent activity is an activity that can be retried. Keeping this parameter's value as <code>True</code> allows idempotent activities by preventing the BPEL server from dehydrating immediately after a failed activity. This parameter is configured in a partner link at runtime in BPEL.
validateXML Default: False	False means that the system will not validate all XML messages during a receive activity. This parameter is configured in a partner link at runtime in BPEL.

12.3 Using Other Tuning Strategies

Once you have tuned the parameters listed in the previous section, you can consider using the following strategies to further improve performance.

This section includes the following topics:

- [Identifying Tables Impacted By Instance Data Growth](#)
- [Troubleshooting Frequent Transaction Timeouts While Displaying Composite Instances](#)

12.3.1 Identifying 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.

You can use [Table 12–5](#) to identify tables that may be affected by instance data growth. See "Monitoring Space Usage, Hardware Resources, and Database Performance" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for advice on how to monitor performance on the following database tables:

Table 12–5 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.
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 <code>audit_trail</code> 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 <code>audit_trail</code> 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 <code>onMessage</code> activity) a subscription is written out for that specific receive activity.
document_ci_ref	Stores cube instance references to data stored in the <code>xml_document</code> table.

Table 12–5 (Cont.) Oracle BPEL Process Manager Tables Impacted by Instance Data Growth

Table Name	Table Description
document_dlv_msg_ref	Stores references to dl_v_message documents stored in the xml_document table.
wftask	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, dl_v_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.
Headers_properties	Stores headers and properties information.

When you have determined which tables are causing slow performance, you can purge them. See "Understanding Growth Management Challenges and Testing Strategies" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite* for more information on managing database growth.

12.3.2 Troubleshooting Frequent Transaction Timeouts While Displaying Composite Instances

If you are experiencing transaction timeouts while fetching composite instances from Enterprise Manager, you may need to add a FLOW_ID-based index to the CUBE_INSTANCE schema table.

This index is required for Enterprise Manager searches that retrieve all the component instances associated with a given Flow ID. This index may be missing in a fresh 12c (12.1.3) installation or in an upgraded environment.

To fix this in a cluster environment with several Managed Servers, you would do the following:

1. Stop the Admin and Managed Servers if they are running.
2. Connect to the SOAINFRA schema using SQL.
3. Run the following command:

```
CREATE INDEX CI_FLOW_ID ON CUBE_INSTANCE(FLOW_ID);
```
4. Restart the Admin and Managed Servers.

Tuning Oracle Mediator

This chapter describes how to tune Oracle Mediator for improved performance.

It contains the following topics:

- [Section 13.1, "About Oracle Mediator"](#)
- [Section 13.2, "Tuning Mediator Parameters"](#)

13.1 About Oracle Mediator

Mediator is a component of the Oracle SOA Suite 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 *Developing SOA Applications with Oracle SOA Suite*.

For details about Oracle Mediator, see "Administering Oracle Mediator Service Components and Engines" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

13.2 Tuning Mediator Parameters

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 under "Administering Oracle Mediator Service Components and Engines" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

[Table 13–1](#) describes the parameter values that can be tuned for performance. Note that the need to tune Mediator to improve performance is unlikely.

Table 13–1 Essential Mediator Tuning Knobs

Parameter	Tuning Recommendation
DeferredMaxRowsRetrieved Default: 20 rows	Increase the default value to retrieve more deferred processing messages from the DB in one iteration. Note that in Mediator, this parameter is only used with parallel routing rules.
DeferredLockerThreadSleep Default: 2 seconds	If deferred messages constitute a small percentage of total messages, increase the default value to perform fewer trips to the DB to retrieve deferred messages. Some use case scenarios can benefit from an idle time of 3600 seconds (60 minutes).
metricsLevel Default: enabled	If you do not need to collect DMS metrics data, disabling this parameter can improve performance.

For more information about each parameter, see "Configuring Oracle Mediator Service Components and Engines" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

13.2.1 Using Resequencer for Messages

A Resequencer is used to rearrange a stream of related but out-of-sequence messages back into order. It sequences the incoming messages that arrive in a random order and then sends them to the target services in an orderly manner.

Table 13–2 lists tunable parameters for Resequencer in Mediator. You can tune the following parameters by accessing the Mediator Service Engine Properties page or the System MBean Browser using one of the methods described under "Configuring Oracle Mediator Service Engine Properties" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

Table 13–2 Essential Tuning Knobs for Resequencer in Mediator

Parameter	Tuning Recommendation
ResequencerMaxGroupsLocked Default: 4 rows	Increase the default value to lock more Resequencer groups from the database in one iteration.
ResequencerLockerThreadSleep Default: 10 seconds	Increase the default value to perform fewer trips to the database to lock resequencer groups if resequencer groups constitute a small percentage of total groups and messages.
DeleteMessageAfterComplete Default: True	Keep True as the value to delete message after successful execution. For a high load use case, this will result in more database space. Changing the default value to False will retain the resequenced messages in the resequencer database. This will slow down the resequencer database queries which in turn degrade the performance.

For more information about Resequeencers in Mediator, refer to "Configuring Resequenced Messages" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

Tuning Oracle Managed File Transfer

This chapter describes how to tune Managed File Transfer (MFT) for performance or for troubleshooting.

This chapter contains the following sections:

- [About Managed File Transfer](#)
- [Tuning MFT Parameters](#)

14.1 About Managed File Transfer

Oracle Managed File Transfer (MFT) is a high performance, standards-based, end-to-end managed file gateway. It features design, deployment, and monitoring of file transfers using a lightweight web-based design-time console that includes file encryption, scheduling, and embedded FTP and sFTP servers.

For more information about Managed File Transfer, see "Understanding Oracle Managed File Transfer" in *Using Oracle Managed File Transfer*.

14.2 Tuning MFT Parameters

This section describes the parameters to optimize performance. It includes the following topics:

- [Tuning Remote FTP / SFTP / FILE Type Sources](#)
- [Minimizing MDS label](#)
- [Adjusting the Materialized Views Refresh Interval](#)

[Table 14–1](#) lists and describes knobs that you will likely need to tune to improve MFT performance. To diagnose problem areas in MFT, see "Monitoring Oracle Managed File Transfer" and "Administering Oracle Managed File Transfer" in *Using Oracle Managed File Transfer*.

Table 14–1 Essential MFT Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
Processor count Default: 2 for each type of processor	JMS messages are accumulating in message processing queues.	Increase the processor count for the queues where messages are accumulating. The optimal value depends on the meta data and incoming payload. You can calculate the optimal processor count using DMS metrics. To enable DMS metrics, add MBean property <code>enablePerformanceMetric</code> . To disable metrics later, set the value to <code>False</code> .	Having more processors requires more system resources for concurrent processing.
Maximum Concurrent Request and Max Logins settings for Embedded FTP/SFTP server Default: 10	<ul style="list-style-type: none"> ■ Multiple connection requests in waiting status ■ The message "Too many users logged in, user will be disconnected" occurs in the embedded server log file 	Increase the maximum number of concurrent requests and maximum number of logins for embedded FTP/SFTP server. You can increase the count so long as performance continues to scale linearly. If the embedded server service (FTP/SFTP) is not being used, then disable this setting.	Increased count requires more system resources for concurrent processing.
LDAP Max Pool Default: 10	<ul style="list-style-type: none"> ■ Number of concurrent connections to the LDAP consistently reaches max limit 	Increase count. Because LDAP is a shared resource for all deployed applications in WebLogic server, you should monitor LDAP connections and adjust this value accordingly.	Increased count requires more system resources.

Table 14–1 (Cont.) Essential MFT Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
Max connections to MFTDataSource Default: 50	<ul style="list-style-type: none"> Number of concurrent connection to the data source consistently reaches max limit 	<p>Increase the connection count so long as performance continues to scale linearly.</p> <p>Optimal value can be determined based on the number of processors, listening source threading model and max concurrent request settings of embedded servers.</p>	Increased count requires more system resources.
Generating checksum setting Default: Enabled	<ul style="list-style-type: none"> Overall MFT message processing is slow 	Disable this parameter if checksum validation for delivered payloads from MFT is not necessary.	Generating checksum is a time consuming operation.
Regular purge Default: Disabled	<ul style="list-style-type: none"> Disk space is approaching the maximum limit Table space used by MFT tables reaches the max table space allotted 	Run purge to free disk or table space.	Historical information/data will be discarded.

Table 14–2 describes the tuning properties that do not regularly need to be tuned. Keeping their default values is recommended, so you can check these parameters see if their values have been changed.

Table 14–2 MFT Parameters with Low or Medium Importance

Parameter	Problem	Tuning Recommendation	Trade-offs
Processing function/callout usage recommendation for broadcasting use cases Default: Target Level	Associated processing function/callout will be executed for each target which will degrade the performance.	For broadcasting use cases, associate processing functions/callouts at source level instead of target level as much as possible.	None.
Sub-folder count MBean setting Default: 256	Degraded disk performance caused by MFT switching among a high number of sub-folders to store files.	Reduce the sub-folder count.	Reducing the number of sub-folders will increase the number of files stored in each sub-folder. If the volume of incoming files is high, the number of the files inside a single sub-folder will degrade performance.
Store Inline payload setting Default: File System	Slow performance because accessing inline payload for Web Service sources from the disk takes too much time.	Store inline payload in the database rather than the file system.	The table size used by MFT will increase as inline payloads are stored in the database.

Table 14–2 (Cont.) MFT Parameters with Low or Medium Importance

Parameter	Problem	Tuning Recommendation	Trade-offs
"Always Save Modified Files" setting at the target level Default: <code>False</code>	If you have changed this setting to <code>True</code> for auditing purposes, you will have increased disk space usage.	The default value of <code>False</code> reduces disk space usage.	No audit information will be available. Note that a target level resubmit will not work if there was any pre-processing associated with the target.
<code>minFileSizeForProgressMonitor</code> Default: 10 MB	Frequent updates about byte transfer.	Specify a minimum file size so that the transfer progress screen appears for larger files only.	For files smaller than the minimum specified, the file transfer progress will not be displayed.
<code>progressMonitorTimeToCommit MBean</code> Default: 4 seconds	Frequent updates about byte transfer.	Specify a minimum file size so that the transfer progress screen appears for larger files only.	Database updates on bytes transferred for ongoing file transfers will be slower.
<code>MaxMdsSessionCacheCount</code> Default: 100	Out-of-memory exceptions caused by MDS cache memory footprint.	Decrease this value.	Decreasing this will decrease the performance of the overall MFT message processing because accessing data from the cache is faster.

14.2.1 Tuning Remote FTP / SFTP/ FILE Type Sources

If MFT is not able to pick up files even after polling frequency is expired, you need to tune the remote FTP/SFTP/FILE type sources. MFT uses the JCA Adapters underneath for all of these source types. Refer to the SOA adapter recommendations listed under "Oracle JCA Adapter Framework Performance and Tuning" in *Understanding Technology Adapters*.

Some of the properties are listed in [Table 14–3](#).

Table 14–3 Tuning Remote FTP/SFTP/FILE Type Source

Parameter	Problem	Tuning Recommendation	Trade-offs
<code>ThreadCount</code> Default: -1	A high priority endpoint is downloading files slowly because of insufficient threads in the global pool.	Specify a value greater than 0. This will create a dedicated thread pool for a given end point to download files.	A very high value may result in lots of threads assigned to one end point, which can lead to lower overall performance.
<code>SingleThreaded</code> Default: <code>False</code>	In rare cases, you may not want to use global threads or allocate a separate thread pool for a low-priority end point.	Set value to <code>True</code> .	If set to true, it can result in a delay in downloading files from the end point as now there will be a single thread for polling as well as downloading new files.

14.2.2 Minimizing MDS label

Artifact deployment results in creation of new MDS labels. More MDS labels will increase the memory footprint and time to retrieve the metadata.

In general, users should follow these best practices for deployments:

- Minimize frequent deployments and meta data creations.
- Use bulk deployment WLST commands.

- Make all changes for metadata and deploy them at once.

14.2.3 Adjusting the Materialized Views Refresh Interval

Materialized views refresh every 1 minute. If there is a heavy load on the database server, you may want to increase the refresh frequency from 1 minute.

You can view data from materialized views on the MFT console. If a high load is observed on the database server, this refresh frequency can be adjusted using the following command:

```
ALTER MATERIALIZED VIEW <<MV_NAME>> REFRESH NEXT <<REFRESH_INTERVAL>>;
```

Below are the materialized views used by MFT:

- MV_MFT_PAYLOAD_INFO
- MV_MFT_SOURCE_INFO
- MV_MFT_SOURCE_MESSAGE
- MV_MFT_TARGET_INFO
- MV_MFT_TRANSFER
- MV_MFT_TRANSFER_COUNT_INFO

Tuning Oracle Business Rules

This document explains how to tune performance for Oracle Business Rules. Oracle Business Rules technology enables automation of business rules; it also enables extraction of business rules from procedural logic such as Java code or BPEL processes.

This chapter includes the following sections:

- [Section 15.1, "About Oracle Business Rules"](#)
- [Section 15.2, "Tuning Oracle Business Rules"](#)

15.1 About Oracle Business Rules

Oracle Business Rules provides an easy-to-use authoring environment as well as a very high-performance inference-capable rules engine. Oracle Business Rules is part of the Oracle Fusion Middleware stack and will be a core component of many Oracle products including both middleware and applications.

For more information, see the *User's Guide for Oracle Business Rules* and "Getting Started with Oracle Business Rules" in *Developing SOA Applications with Oracle SOA Suite*.

15.2 Tuning Oracle Business Rules

This section describes recommended tuning strategies that may optimize performance.

In most cases, writing of Rules should not require a focus on performance. However, as in any technology, there are tips and tricks that can be used to maximize performance when needed. Most of the considerations are focused on the initial configuration of the data model.

Table 15–1 Essential Business Rules Tuning Strategies

Strategy	Description	Recommendation
Use Java Beans	The rule engine is most efficient when the facts it is reasoning on are Java Beans (or RL classes) and the associated tests involve bean properties.	The beans should expose get and set methods (if set is allowed) for each bean property. If application data is not directly available in Java Beans, flatten the data to a collection of Java Beans that will be asserted as facts (and used in the rules).
Assert Child Facts instead of Multiple Dereferences	Expressions like <code>Account.Contact.Address</code> involve more than one object dereference. In a rule condition, this is not as efficient as expressions with single dereferences.	It is a best practice to flatten fact types as much as possible. If the fact type has a hierarchical structure, consider using <code>assertXPath</code> or other means to assert object hierarchy.
Avoid Side Effects in Rule Conditions	The tests in a rule condition may be evaluated a greater or lesser number of times than would occur in a procedural program.	Methods or functions that have side effects such as changing a value or state should not be used in a rule condition. If a method or function has side effects, those side effects may be performed an unexpected number of times.
Avoid Expensive Operations in Rule Conditions	Expensive operations would include any operation that involves I/O (disk or network) or even intensive computations. These operations should be done external to the rules engine.	Expensive operations should be avoided in rule conditions. In general, consider avoiding I/O or DBMS access from the rules engine directly. For other expensive operations or calculations, consider performing the computations and assert the results as a Java or RL fact. These facts are used in the rule conditions instead of the expensive operations.
Consider Pattern Ordering	Reordering rule patterns can improve the performance of rule evaluation in time, memory use, or both. Finding the optimal order for your system will require some experimentation.	If a fact is not expected to change (or will not change frequently) during rule evaluation, order the fact clauses by expected rate of change from least to greatest. If a fact clause (including any tests that involve only that fact) is expected to match fewer facts than other fact clauses in the rule condition, order the fact clauses from most restrictive (matches fewest facts) to least restrictive.
Consider the Ordering of Tests in Rule Conditions	Proper ordering can reduce the amount of computation required for facts that do not satisfy the rule condition.	The tests in a rule condition should be ordered so that a more restrictive test occurs before a less restrictive test. If the degree of restrictiveness is not known, or estimated to be equal for a collection of tests, then simpler tests should be placed before more expensive tests.

Tuning Oracle Business Process Management

The Oracle Business Process Management Suite provides a seamless integration of all stages of the application development life cycle from design-time and implementation to run-time and application management.

This chapter contains the following sections:

- [Section 16.1, "About Oracle Business Process Management"](#)
- [Section 16.2, "Tuning Business Process Management Parameters"](#)
- [Section 16.3, "Using Other Tuning Strategies"](#)

16.1 About Oracle Business Process Management

The Oracle Business Process Management Suite provides an integrated environment for developing, administering, and using business applications centered around business processes. Oracle Business Process Management is layered on the Oracle SOA Suite and shares many of the same product components, including Business Rules, Human Workflow, and Oracle Adapter Framework for Integration.

For more information on using Oracle Business Process Management, see the *Oracle Fusion Middleware User's Guide for Oracle Business Process Management*.

For more information on tuning Oracle Business Process Management with your other Oracle Fusion Middleware components, see [Chapter 2, "Top Performance Areas"](#).

16.2 Tuning Business Process Management Parameters

This section describes the parameters you should tune when your primary objective is optimizing performance. These parameters can be tuned in the Enterprise manager, via the SOA Administration in BPMN properties.

To tune the performance of the Oracle Business Process Management engine, you can reduce resource demands to reduce latency and increase system capacity to enable greater scalability.

To reduce resource demands, you can tune the parameters listed in [Table 16-1](#):

Table 16–1 Essential Business Process Management Tuning to Reduce Resource Demands

Parameter	Problem	Tuning Recommendation	Trade-offs
largedocumentthres hold Default: 10000 (100 kilobytes)	Instances are being processed slowly because you are storing large BPMN Data objects.	Decrease the maximum size (in kilobytes) of this parameter to limit the size of BPMN Data Objects. If they surpass this limit, they are stored in a separate location from the rest of the instance scope data. This property is applicable to both durable and transient processes.	The overflow data will be stored in an external append-only table. This adds to overall database size and can increase the overall workload when loading instances from the database.
auditLevel Default: Inherit from Infrastructure	You are seeing frequent database inserts into the audit_trail table. These are caused by audit events being logged by a process.	Reduce or disable audit. You can switch to any of the following settings: <ul style="list-style-type: none"> ▪ Off to log no events or audit events ▪ Minimal to log only events ▪ Error to log only serious problems You can also consider expanding the size of the AuditKeyExtents.	You lose granular error reporting that you could use to diagnose problems later. Always choose the audit level according to your business requirements and use cases. For more information on how to use audit trails for monitoring, see "Monitoring BPMN Process Service Components and Engines" in <i>Administering Oracle SOA Suite and Oracle Business Process Management Suite</i> .

You can also try to purge completed instances as allowed by business requirements and add indexes for any flex fields.

Increasing system capacity should only be done based on analysis of performance testing results. Adjustments should not be made unless testing indicates that they are a constraint on scale. To increase system capacity, you can tune the parameters listed in [Table 16–2](#):

Table 16–2 Essential Business Process Management Tuning to Increase System Capacity

Parameter	Problem	Tuning Recommendation	Trade-offs
DispatcherEngineThreads Default: 2 threads	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. You may need to tune this parameter if you have either of the following problems: You do not have enough threads. CPU utilization remains low OR You have too many threads. Context switching drives up CPU use but does not increase throughput.	Increase the thread pool sizes for the respective workload based on thread pool usage. If you see a backlog of requests and additional CPU capacity, then consider increasing the number of threads.	Increasing the size of the thread pools will increase CPU utilization and garbage collection. However, if there are other application that share the same platform and expect to have access to additional CPU capacity, then that could cause a conflict.
DispatcherInvokeThreads Default:20 threads	Invocation dispatcher messages are generated for each payload received and are meant to instantiate a new instance. You may need to tune this parameter if you have either of the following problems: Threads are busy but CPU utilization remains low. OR You have too many threads. Context switching drives up CPU use but does not increase throughput.	Increase the thread pool sizes for the respective workload based on thread pool usage. If you see a backlog of requests and additional CPU capacity, then consider increasing the number of threads.	Increasing the size of the thread pools will increase CPU utilization. However, if there are other application that share the same platform and expect to have access to additional CPU capacity, then that could cause a conflict.

You can also try to increase the size of connection pools and optimize the use of connections.

16.3 Using Other Tuning Strategies

Once you have tuned the parameters listed in the previous section, consider using the following strategies to further improve performance.

16.3.1 Tuning Oracle Workspace Applications

Database performance and session state management are the primary drivers for performance. Effective database tuning and configuration of HTTP session timeout are important.

Application design is the next largest factor, especially if there are additional data controls used to render contextual data on task forms. In those cases, it is important to optimize data access from those data controls and when possible defer retrieving additional data unless it is needed. For more details on tuning ADF, see [Section 9.2.1, "Oracle ADF Faces Configuration and Profiling"](#).

The following parameters can be changed in the web.xml descriptor in the OracleBPMWorkspace web application. Once they have been modified, you may have to redeploy.

Parameter	Problem	Tuning Recommendation	Trade-offs
HTTP Session Timeout Default: 15 minutes	Memory is being allocated for users who may no longer be actively using the system.	To better manage resource usage, decrease the session timeout value, in minutes, to the smallest value that preserves the expected user experience. This allows the system to reclaim any resources associated with unused sessions as soon as possible. This parameter is edited in the in the web.xml file. The following is a sample snippet of web.xml: <pre><session-config> <session-timeout> 5 </session-timeout> </session-config></pre>	A short timeout value may mean users will have to login ore often if they let the time expire. They also may potentially lose session data.
ADF Client State Token Default: 15	The default value may consume too much memory.	Decrease the value to 3 in order to minimize the memory footprint. Through this setting, you can control the number of pages users can navigate using the browser Back button without losing information. To reduce CPU and memory usage, you can decrease the value in the web.xml file. The following is a sample snippet of web.xml: <pre><context-param> <param-name> org.apache.myfaces.trinidad.CLIENT_ STATE_MAX_TOKENS </param-name> <param-value> 3 </param-value> </context-param></pre>	If the user clicks on the 'Back' button more than 3 times, there will be no session data stored for that page. If the value is too small, users will get an error using the back button.

Parameter	Problem	Tuning Recommendation	Trade-offs
Compress_View_State Token Default: True	Slow performance on slower/higher latency networks.	Keep the default to keep zipping enabled. This setting controls whether or not the page state is compressed. Zipping greatly reduces the memory being taken up by page state in the session object. The following is a snippet of the web.xml: <param-name>org.apache.myfaces.trinidad.COMPRESS_VIEW_STATE</param-name> <param-value> true </param-value>	There is an additional CPU cost to zipping and unzipping the view state.
DISABLE_CONTENT_COMPRESSION Default: False	Slow initial load of pages.	In production environments, make sure you remove the DISABLE_CONTENT_COMPRESSION parameter from the web.xml file or set it to FALSE. By default, style classes that are rendered are compressed to reduce page size. The following is a snippet of the web.xml: <param-name>org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION</param-name> <param-value> false </param-value>	None.

16.3.2 Process Measurement

Process Analytics uses measurement events to sample the process and publish measurements to registered consumers. In 12c (12.1.3), these measurements can be enabled by setting the `DisableAnalytics` parameter to `False` in the BPM Enterprise Manager's Analytics Configuration MBean.

The two supported consumers for measurements in 12c are BAM 11g Monitor Express and BAM 12c Process Metrics. They can be enabled or disabled using the `DisableProcessMetrics` and `DisableMonitorExpress` attributes of the `AnalyticsConfig` mbean.

Note: Only data that is useful should be published. The process design specifies what data (dimensions, measure, counters) should be published and at what point(s). If data is being generated that is not useful, then it could be adding unnecessary load to the system.

Measurement events are published on the JMS Topic: `MeasurementTopic`, and consumed by registered Action MDBs. In order to tune JMS for Measurements, consider changing the parameters listed in [Table 16-3](#), as needed, in a high volume environment:

Table 16–3 Essential JMS Resource Tuning for BPM

JMS Resource	Problem	Tuning Recommendation	Trade-offs
dist_MeasurementTopic_auto Default: Forwarding Policy Replicated	A distributed measurement topic in a cluster installation is configured by default with FORWARDING POLICY REPLICATED even though this is not the best performance option for BPM analytics.	Change the Forwarding Policy for this parameter to PARTITIONED. This parameter can be altered in the WebLogic console. You can find it from the front page with the following options: JMS Modules -> BPMJMSModule -> dist_MeasurementTopic_auto. You will need to restart all SOA BPM cluster nodes for the changes to take effect.	A distributed topic with a Partitioned policy generally outperforms the FORWARDING POLICY REPLICATED. For more information on distributed topics versus other topic types, see "Supported Topic Types" in <i>Developing Message-Driven Beans for Oracle WebLogic Server</i> . For more information on partitioned and replicated forwarding policies, see "Configuring Partitioned Distributed Topics" in <i>Administering JMS Resources for Oracle WebLogic Server</i> .
MeasurementTopicConnectionFactory Default: Send Timeout 200000	You have a high volume environment and you are receiving frequent resource allocation exceptions from message producers. For more information, see "Defining a Send Timeout on Connection Factories" in <i>Tuning Performance of Oracle WebLogic Server</i> .	Increase the Send Timeout for this parameter to 240000 in a high volume environment. The numerical value represents the maximum length of time in milliseconds. This parameter can be altered in the WebLogic console. You can find it from the front page with the following options: JMS Modules -> BPMJMSModule --> MeasurementTopicConnectionFactory --> Default Delivery.	You may create a message backlog that consumes memory and resources.
MeasurementQuota Defaults: Message Maximum 1000000 and Bytes Maximum 800000000	Measurement messages cannot be published and fails with javax.jms.ResourceAllocationException thrown.	Set the Message Maximum and Bytes Maximum for this parameter equal to the amount of system memory available after you have accounted for the rest of your application load. MeasurementQuota attributes can be altered in the WebLogic console. You can find it from the front page with the following options: JMS Modules -> BPMJMSModule -> MeasurementQuota.	Increasing this value consumes more memory. Message delivery may still fail if the aggregate size of messages pushed to the consumer is larger than the current protocol's maximum message size. For more information about measurement quotas, see "Tuning for Large Messages" in <i>Tuning Performance of Oracle WebLogic Server</i> .
BPMJMSServer Default: MessageBuffer size 100000	The JMS server is frequently writing message bodies to disk.	Increase the Message Buffer Size for a given BPMJMSServer. This will Note that the BPMJMSServer uses Paging File and JMSFileStore. This parameter can be altered in the WebLogic console. You can find it from the front page with the following options: JMS Servers_auto_number.	The JMS server will use more memory.

Tuning Oracle Human Workflow

This chapter describes how to tune Oracle Human Workflow for optimal performance.

You can tune Oracle Human Workflow in these areas:

- [Section 17.1, "About Oracle Human Workflow"](#)
- [Section 17.2, "Tuning Human Workflow"](#)
- [Section 17.3, "Using Other Tuning Strategies"](#)

17.1 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 *Developing SOA Applications with Oracle SOA Suite*.

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

17.2 Tuning Human Workflow

This section discusses how to optimize taskflow. The following suggestions are all applicable to API usage.

Table 17-1 Essential Human Workflow Tuning Strategies

Name	Description	Recommendation
Minimize Client Response Time	<p>Since workflow client applications are interactive, it is important to have good response time at the client.</p> <p>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.</p>	Review your performance metrics to determine how response time can be improved.
Choose the Right Workflow Service Client	<p>Remote client 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.</p> <p>SOAP client is preferred for standardization (based on web services). There are additional performance considerations 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.</p>	<p>If the client application is based on Java EE technology, then consider which client should be used based on your use case scenarios.</p> <p>Note that if the client application is based on .Net technologies, then only the SOAP workflow services can be used.</p>
Narrow Qualifying Tasks Using Precise Filters	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.	Use precise filters to improve response time.
Retrieve Subset of Qualifying Tasks (Paging)	The query API has paging parameters that control the number of qualifying rows returned to the user and the start row.	Decrease the <code>startRow</code> and <code>endRow</code> parameters to values that may limit the number of returned records. This will decrease query time, the application process time, and the amount of data returned to client.
Fetch Only the Information That Is Needed for a Qualifying Task	Typically only some of the payload fields are needed for displaying the task list.	<p>When using the <code>queryTask</code> service, consider reducing the amount of optional information retrieved for each task returned in the list.</p> <p>In rare cases where the entire payload is needed, then the payload information can be requested.</p>
Reduce the Number of Return Query Columns	When using the <code>queryTask</code> service, consider reducing the number of query columns to improve the SQL time.	Try using the common columns as they are the most likely indexed columns. This allows the SQL to execute faster.
Use the Aggregate API for Charting Task Statistics	Sometimes it is necessary to display charts or statistics to summarize task information.	Consider using the new aggregate APIs to compute the statistics at the database level rather than fetching all the tasks using the query API and computing the statistics at the client layer.

Table 17–1 (Cont.) Essential Human Workflow Tuning Strategies

Name	Description	Recommendation
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.	Call the <code>countTasks</code> API method, which returns only the number of matching tasks.
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.	Create indexes on these columns in certain cases where certain mapped flexfield columns are frequently used in query predicates.
Use the <code>doesTaskExist</code> Method	Sometimes it is necessary to check whether a task exists that matches particular query criteria.	Consider using <code>doesTaskExist</code> instead of the default of <code>countTasks</code> . The <code>doesTaskExist</code> 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 <code>countTasks</code> method.

17.3 Using Other Tuning Strategies

Once you have tuned the parameters listed in the previous section, you can consider using the following strategies to further improve performance.

17.3.1 Improving Server Performance

Server performance essentially determines the scalability of the system under heavily loaded conditions. In [Section 17.2, "Tuning Human Workflow"](#), strategy "Minimize Client Task 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 on the server and can improve server performance. In addition, a few other configuration changes can be made to improve server performance:

Table 17-2 Essential server performance tuning strategies

Name	Description	Recommendation
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.	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.
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 task completion.	Ensure that there are sufficient resources to update the external system after the task is completed instead of after every workflow event. 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.
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.	Minimize the notification to alert a user only when a task is assigned instead of sending out notifications for each workflow event. Also 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.
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.

17.3.2 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 decrease the amount of time it takes to complete workflows. Some of these options are discussed in this section:

Table 17–3 Essential workflow completion tuning strategies

Name	Description	Recommendation
Use Workflow Reports to Monitor Progress	Several workflow reports (and corresponding views) are available that can make monitoring and proactive problem fixing easier.	By checking the 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 their tasks.
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.
Specify User and Group Rules for Automated Assignment	Rules can help significantly reduce workflow waiting time, which results in faster workflow completion.	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.
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.	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.

17.3.3 Tuning the 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 the user can 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.

- If using Oracle Internet Directory as a Identity Provider, ensure that you run the `oidstats.sql` to gather latest statistics on the database after the data shape has changed.

17.3.4 Tuning the Database

The Human Workflow schema is shipped with several indexes defined on the most important columns. 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, "Tuning Database Parameters"](#).

Tuning Oracle Business Activity Monitoring

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 includes the following sections:

- [Section 18.1, "About Oracle Business Activity Monitoring"](#)
- [Section 18.2, "Tuning BAM Server Parameters"](#)

18.1 About Oracle Business Activity Monitoring

Oracle Business Activity Monitoring (BAM) provides the tools for monitoring business services and processes in the enterprise. It allows correlation 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 information on how to monitor your BAM installation's performance, see "Monitoring Oracle BAM Performance" in *Monitoring Business Activity with Oracle BAM*.

18.2 Tuning BAM Server Parameters

BAM performance largely depends on the performance of the following components:

- The Weblogic Server. See *Tuning Performance of Oracle WebLogic Server*.
- Metadata Service. See [Chapter 7, "Tuning Oracle Metadata Service"](#).
- Coherence. See *Administering Oracle Coherence*.
- ADF. See [Chapter 9, "Tuning Oracle Application Development Framework \(ADF\)"](#).
- Database Settings. See [Chapter 2.6, "Tuning Database Parameters"](#).
- Java Virtual Machines (JVMs). See "Tuning Java Virtual Machines (JVM)" in *Tuning Performance of Oracle WebLogic Server*.
- Oracle Platform Security Service. See ["Tuning Oracle Fusion Middleware Security Performance Tuning"](#).

BAM performance also depends on good data object design strategies at design time and on having good data object purging strategies at runtime.

While BAM 12c can support much larger transaction volumes (data arrival rates into BAM), BAM 12c is an operational analytics product, not a business intelligence product.

Hence, it is recommended that data that is of analytical value for operational decision-making be kept in BAM. For most customers, this means storing about 5-30 days of transactional data in BAM. Resting data sizes typically comparable to a data warehouse are not useful for operational decision-making, so such data volumes do not constitute a mainstream use case for BAM 12c.

The tuning suggestions listed and described in [Table 18-1](#) can be used to improve performance of the BAM Server:

Table 18-1 Essential BAM Server Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
Max connections to BAMDataSource Default: 50	The number of concurrent connection to the data source consistently reaches max limit.	Increase count so long as performance continues to scale linearly. This is set at the WebLogic level. The value can be determined mainly based on the number of processors, listening source threading model and max concurrent request settings of embedded servers.	Increasing the count will most likely increase the system resources usage.
Viewset Expiry Time Default: 180 seconds	Viewsets are lingering after the DC connection is lost.	Decrease the expiry time value so that viewsets do not linger. See "Monitoring Viewsets" in <i>Monitoring Business Activity with Oracle BAM</i> for information on how to find and modify this parameter.	None.
DiagnosticLevel Default: Info	You need granular diagnostic logs to identify a problem. OR Your system is running fine and you do not need detailed logs.	Keeping the default of INFO will help performance. For more information on using the BAM Diagnostic Framework, see "Using the BAM Diagnostic Framework" in <i>Monitoring Business Activity with Oracle BAM</i> .	You will not have detailed logs to identify a problem if your system slows down.
ASM (Automatic Server Migration) Default: WSM	You want to migrate a SOA Suite installation with BAM to High Availability. Because BAM is a real-time system, you should enable ASM.	ASM is used so High Availability can occur faster than WSM. Given that BAM is a real time system, ASM is required for BAM HA	None.
JVM heap size Default: -Xms768m -Xmx1536m	Oracle BAM is running slowly and out-of-memory exceptions occur.	Increase the heap size to 2 GB. Use the following command with the -Xms2048m and -Xmx2048m arguments: setenv USER_MEM_ARGS "-Xms2048m -Xmx2048m -XX:PermSize=256m -XX:MaxPermSize=768m"	Increasing the JVM heap size for BAM could affect other SOA components. For more heap size tuning tips, see, "Tuning Tips for Heap Sizes" in <i>Tuning Performance of Oracle WebLogic Server</i> .

18.3 Other Tuning Strategies

If Oracle BAM is running more slowly than expected, you can try the following strategies:

- [Creating an Index Column](#)
- [Tuning Loggers](#)
- [Tuning Continuous Query Service](#)

18.3.1 Creating an Index Column

If throughput of data into a data object from an Enterprise Message Source or other source is slow, create an index column for the primary key column. See "Adding Index Columns" in *Monitoring Business Activity with Oracle BAM* for more information.

18.3.2 Tuning Loggers

The default Oracle Diagnostic Logging Level for all loggers is `Notification`. For stress testing and production environments, consider using the lowest acceptable logging level, such as `ERROR` or `WARNING`.

The loggers in BEAM that can affect BAM performance are as follows:

```
oracle.beam.common.alertsengine
oracle.beam.server.service.alertsengine
oracle.beam.Common
oracle.beam.cqservice
oracle.beam.composer
com.oracle.beam
oracle.beam.datacontrol
oracle.beam.datacontrol.management
oracle.beam.server.service.ems
oracle.beam.messaging
oracle.beam.server.service.persistence
oracle.beam.server.service.reportcache
oracle.beam.security
oracle.beam.mbean
oracle.beam.shared
oracle.beam.server
oracle.beam.impexp.t2p
oracle.beam
```

For information about locating these loggers and changing their Oracle Diagnostic Logging Level, see "Configuring Log Files" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

18.3.3 Tuning Continuous Query Service

The Continuous Query Service (CQS) is a BAM-specific wrapper around the Continuous Query Language (CQL) engine within the Oracle Complex Event Processing Service Engine. The CQS is a pure push system: query results are delivered automatically. The CQS supports both stream (non-persistent) and archived relation (persistent) data objects.

When you create a query, the CQS sets up tables in the CQL engine, registers the query, and listens for data changes from the persistence engine. The query result is processed in the CQL engine, then pushed to the CQS and on to the report cache.

For information on how to monitor continuous queries for performance issues, see "Monitoring Continuous Queries" in *Monitoring Business Activity with Oracle BAM*. Once you understand how your current system is performing, you can try to improve performance by tuning the knobs described in [Table 18–2](#). Note that for most of these parameters, tuning for performance means losing diagnostic information.

Table 18–2 Tuning the Continuous Query Service

Parameter	Problem	Tuning Recommendation
Data Object type Default: None	You have arbitrarily designated simple data objects as stream, archived stream, and archived relation, and are not sure what to do.	Categorize your data objects as stream if you do not care about history data. See "Data Object Types" in <i>Monitoring Business Activity with Oracle BAM</i> for detailed descriptions of each data object type and relation.
Data Object purging Default: Disabled	By default, data object retention is not set. Many rows in data object will cause performance issue.	Customer can set Data Object retention to in Data Object Retention tab to specify how many days they want to keep the data in a Data Object. When the specified days has elapsed, the data rows will automatically be purged. See "Setting Data Retention" in <i>Monitoring Business Activity with Oracle BAM a Data Object</i> in for information on how to find and change this setting.
Replay for Archived Stream Data Objects	Data parsing is slow for archived stream data objects.	Specify a smaller Replay Unit or a lower Replay Amount to reduce the amount of past data retained in memory. This will reduce the time and memory to parse data retrieved from the database.
Time Window on Input Streams	You have chosen to turn an Active Data query into a continuous query and are receiving out-of-memory exceptions.	Decrease the time window size on the Active Data stream. This restricts the amount of memory the window uses to store elements. To get an idea of how much window size affects memory usage, consider a scenario where the Window Size = 1 hour (RANGE 1 hour) and the event size = 100 bytes. If the event rate is 1000 events / second, then the window will contain 1000 * 3600 events when it is full. The memory consumed is 1000 * 3600 * 100 bytes = ~340 MB. See "Enabling Active Data in a View" in <i>Monitoring Business Activity with Oracle BAM</i> for how to configure the window size on an active data view.

Table 18–2 (Cont.) Tuning the Continuous Query Service

Parameter	Problem	Tuning Recommendation
Active Data Collapsing Interval Default: Unchecked	You have checked the box for Active Data Collapsing to make data aggregation active. You want more frequent snapshots or need to free up memory.	<p>Define a smaller Interval to make the view update more frequently and to reduce the amount of aggregated data stored in memory.</p> <p>You can maximize your memory usage by taking note of the evaluation interval, the event size, and the event rate. Given the following values:</p> <p>Interval: Every 5 minutes Event Size: 100 bytes Event Rate: 1000 events/second</p> <p>The maximum size of the aggregated view is $5 * 60 * 1000 = 300,000$ events = ~28 MB.</p> <p>See "Using Active Data" in <i>Monitoring Business Activity with Oracle BAM</i> for information on finding the Active Data Collapsing setting.</p>
Slow Changing Dimension for Data Object Dimension Tables Default: Unchecked	Continuous queries on dimension tables are slow and consuming memory.	<p>Check this property to activate it. This will indicate that the data in this dimension table changes infrequently.</p> <p>For information on specifying slow-changing dimensions for a data object, see "Specifying Slow-Changing Dimensions for a Data Object" in <i>Monitoring Business Activity with Oracle BAM</i>.</p>
Query Type Default: SQL	You are experiencing out-of-memory exceptions and most of your queries are continuous.	Use schedule query (SQL) where you do not expect frequent output. This saves memory because SQL involves JDBC resources while CQL stores data in memory.

Tuning Oracle Service Bus

This chapter describes tips for tuning Oracle Service Bus performance.

It contains the following sections:

- [Section 19.1, "About Oracle Service Bus"](#)
- [Section 19.2, "Tuning OSB Parameters"](#)
- [Section 19.3, "Using Other Tuning Strategies"](#)

19.1 About Oracle Service Bus

Within a SOA framework, Oracle Service Bus (OSB) provides connectivity, routing, mediation, management and also some process orchestration capabilities. The design philosophy for OSB is to be a high performance and stateless (non-persistent state) intermediary between two or more applications. However, given the diversity in scale and functionality of SOA implementations, OSB applications are subject to a large variety of usage patterns, message sizes and QOS requirements.

In most SOA deployments, OSB is part of a larger system where it plays the role of an intermediary between two or more applications (servers). A typical OSB configuration involves a client invoking an OSB Proxy Service which may make one or more service callouts to intermediate back-end services and then route the request to the destination back end system before responding to the client.

It is necessary to understand that OSB is part of a larger system and the objective of tuning is the optimization of the overall system performance. This involves not only tuning OSB as a standalone application, but also using OSB to implement flow-control patterns such as throttling, request-buffering, caching, prioritization and parallelism.

For more information about Oracle Service Bus, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

19.2 Tuning OSB Parameters

Oracle Service Bus performance largely depends on the performance of the other components. The following components affect OSB performance:

- WebLogic Server
- Coherence
- Adapters

You can begin tuning Oracle Service Bus if you believe the above components are tuned to your satisfaction.

19.2.1 Tuning Oracle Service Bus with Work Managers

Starting in 12c (12.1.3), Oracle Service Bus can be tuned by several Oracle WebLogic Server Work Managers.

For example, Split-Join tuning can be accomplished using Work Managers. By default, applications do not specify a Work Manager for Split-Joins, but Split-Joins can be assigned a Work Manager if there are strict thread constraints that need to be met, such as scheduling parallel tasks.

For optimal performance, strike a balance between the following Work Manager constraints:

- `min-threads-constraint` so that Split-Join operations are not starved of threads.
- `max-threads-constraint` so that Split-Joins do not starve other resources

By default, there is no minimum or maximum thread constraint defined, which could either slow Split-Join operations down or slow down other operations sharing the same thread pool.

Work Managers will take Split-Join operations into account when allotting threads to system-wide processes so that this balance is met automatically.

For more information on tuning OSB with Work Managers, see "Using Work Managers with Oracle Service Bus" in *Developing Services with Oracle Service Bus*.

19.2.2 Tuning OSB Operation Settings

[Table 19-1](#) lists and describes the knobs you will most likely need to tune to improve performance. For more information on monitoring Oracle Service Bus to diagnose trouble areas, see "Monitoring Oracle Service Bus" in *Administering Oracle Service Bus*.

Table 19–1 Essential OSB Operation Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
Monitoring and Alerting Default: Disabled	The Monitoring and Alerting framework is designed to have minimal impact on performance, but all of these processes have performance impacts. In general, the more monitoring rules and pipeline actions you have defined, the larger the performance impact.	Keep the default of Disabled at the OSB level. Most settings can be defined globally or per service. The settings for monitoring and alerting can be configured in the Enterprise Manager Administrator Console. Note that monitoring must be enabled for SLA alerts but not for Pipeline alerts.	Disabling these processes to improve performance means you will be sacrificing certain metrics and alerts that could help you troubleshoot issues in the future. For more information on the OSB Monitoring Framework, see Introduction to the Oracle Service Bus Monitoring Framework in <i>Administering Oracle Service Bus</i> .

Table 19–1 (Cont.) Essential OSB Operation Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
Tracing Default: Disabled	If you have large message sizes and high throughput scenarios, tracing may be slowing your system down.	Leave tracing disabled to improve performance. For more information, see "How to Enable or Disable Tracing" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> .	If disabled, you will lose metrics. Tracing prints the entire message context, including headers and message body. This is an extremely useful feature both in a development and production environment for debugging, diagnosing and troubleshooting problems involving message flows in one or more proxy services.
com.bea.wli.sb.pipeline.RouterRuntimeCache.size Default: 100	You may have one of the following issues: Proxy services are accessed slowly. This means you want to store more proxy services in the static portion of the OSB cache for pipeline service runtime meta-data. The proxy services stored here are never garbage-collected, meaning they are accessed faster. OR You are seeing a lot of cache misses in DMS dumps.	If you want to include more proxy services in the static cache, increase this value as long as there is sufficient memory for runtime data processing for large number of proxy services. If you are seeing cache misses in DMS dumps, decrease this value. This system property caps the number of proxy services in the static portion of the OSB cache for pipeline service runtime meta-data. These services never get garbage collected. You set the size of this value in the setDomainEnv.sh file as an extra java argument, as follows: -Dcom.bea.wli.sb.pipeline.RouterRuntimeCache.size={size} For example, if you want to set this value to 3000, you would write: EXTRA_JAVA_PROPERTIES= "-Dcom.bea.wli.sb.pipeline. .RouterRuntimeCache.size=3000 \${EXTRA_JAVA_PROPERTIES}"	Increasing this value decreases the time it takes to make initial calls to the proxy server. It can also preload the cache when a configuration session is committed. However, while caching proxy services helps reduce compilation costs, it also increases memory consumption. Decreasing this value may mean you will free up memory, but making initial calls to the proxy server may take longer.

Table 19–1 (Cont.) Essential OSB Operation Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
reorderJsonAsPerXmlSchema Default: False	<p>JSON input to REST service may not be ordered as expected by the schema definition.</p> <p>When converting from JSON to XML, OSB runtime uses the order in which JSON name/value appear to construct the corresponding XML element. While well-formed, this format is not valid according to XML schema.</p>	<p>Set this parameter to <code>True</code> by running the REST wizard and checking the box on the first page.</p> <p>Checking this option will make the REST service reorder the input JSON so that the response from the external REST endpoint can be ordered as per the valid schema definition.</p>	Using this option adds significant performance overhead.

19.3 Using Other Tuning Strategies

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.

This section contains the following:

- [Tuning Resequencer in OSB](#)
- [Considering Design Time for Proxy Applications](#)
- [Tuning XQuery](#)
- [Tuning Poller-based Transports](#)

19.3.1 Tuning Resequencer in OSB

A Resequencer is used to rearrange a stream of related but out-of-sequence messages back into order. It sequences the incoming messages that arrive in a random order and then sends them to the target services in an orderly manner.

You can fine-tune the Resequencer by setting the properties listed in [Table 19–2](#) using the Global operational settings page in the OSB EM console:

Table 19–2 Essential Resequencer Tuning

Parameter	Problem	Tuning Recommendation	Trade-offs
ResequencerMaxGroupsLocked Default: 4 groups	<p>This parameter defines the maximum number of message groups that can be locked by resequencer locker threads for parallel processing. The locked groups can then use worker threads to process their respective messages.</p> <p>If message processing is being delayed, identify which of the following situations is true:</p> <ul style="list-style-type: none"> ▪ Incoming messages belong to many groups ▪ There are many messages and they belong to fewer groups 	<p>Increase this parameter's value if you have many groups with a small number of messages each. Resequencer will lock more groups in one iteration.</p> <p>Decrease this value if you have a few groups with many messages. Resequencer will lock less number of groups for processing.</p>	<p>Increasing the MaxGroupsLocked value may result in locking more groups than there are available worker threads. This could result in groups getting blocked while waiting for the availability of the worker threads for message processing.</p> <p>Decreasing the default value may result in under utilization of resources.</p>
ResequencerLockerThreadSleep Default: 10 seconds	<p>The resequencer locker thread queries the database to lock groups for parallel processing. When no groups are available, the locker thread "sleeps" for the configured amount of time specified by this parameter.</p> <p>This parameter needs tuning if you have either of the following situations:</p> <ul style="list-style-type: none"> ▪ You have a high number of messages and processing time between database queries is slow. ▪ You have few messages but frequent database queries. 	<p>Decrease this parameter value if you have a high number of messages to reduce the lag time during processing.</p> <p>Increase this value if Resequencer locker threads are making frequent database round trips even though you do not have many incoming messages.</p>	<p>If the sleep time is too short, there may not be enough worker threads available to process incoming messages of the locked groups. Too many database queries will also cause slow performance.</p> <p>If the time interval between incoming messages is already long, configuring a higher value is not beneficial.</p>
DeleteMessageAfterComplete Default: True	<p>The resequencer database is low on space. If you changed this parameter's value to false, processed messages remain in the resequencer database and slow down database inquiries.</p>	<p>Keep the default value of True to delete message after successful execution. This frees up database space.</p>	<p>You will not have a detailed history of processed messages.</p>

19.3.2 Considering Design Time for Proxy Applications

Consider the design configurations described in [Table 19–3](#) for proxy applications based on your OSB usage and use case scenarios:

Table 19–3 Tuning Design Time for Proxy Application

Strategy	Description	Recommendations
Avoid creating many OSB context variables that are used just once within another XQuery	Context variables created using an Assign action are converted to XmlBeans and then reverted to the native XQuery format for the next XQuery. Multiple "Assign" actions can be collapsed into a single Assign action using a FLWOR expression. Intermediate values can be created using "let" statements.	Avoiding redundant context variable creation eliminates overheads associated with internal data format conversions. This benefit has to be balanced against visibility of the code and reuse of the variables.
Transform contents of a context variable such as \$body.	Transforming the contents of a context variable could be time-consuming.	Use a Replace action to complete the transformation in a single step. If the entire content of \$body is to be replaced, leave the XPath field blank and select "Replace node contents". This is faster than pointing to the child node of \$body (e.g. \$body/Order) and selecting "Replace entire node". Leaving the XPath field blank eliminates an extra XQuery evaluation.
Specify a special XPath.	A general XPath like "\$body/Order" must be evaluated by the XQuery engine before the primary transformation resource is executed. OSB treats "\$body/*[1]" as a special XPath that can be evaluated without invoking the XQuery engine.	Use \$body/*[1] to represent the contents of \$body as an input to a Transformation (XQuery / XSLT) resource. This is faster than specifying an absolute path pointing to the child of \$body.

Table 19–3 (Cont.) Tuning Design Time for Proxy Application

Strategy	Description	Recommendations
Enable Streaming for pure Content-Based Routing scenarios.	OSB leverages the partial parsing capabilities of the XQuery engine when streaming is used in conjunction with indexed XPath. Thus, See Section 19.3.3, "Tuning XQuery" for additional details.	Enabling streaming means that the payload is parsed and processed only to the field referred to in the XPath. Streaming also eliminates the overhead associated with parsing and serialization of XmlBeans. Trade-offs: The gains from streaming can be negated if the payload is accessed a large number of times for reading multiple fields. If all fields read are located in a single subsection of the XML document, a hybrid approach provides the best performance. The output of a transformation is stored in a compressed buffer format either in memory or on disk. Therefore, streaming should be avoided when running out of memory is not a concern.
Set the appropriate QOS level and transaction settings.	OSB can invoke a back end HTTP service asynchronously if the QOS is "Best-Effort". Asynchronous invocation allows OSB to scale better with long running back-end services. It also allows Publish over HTTP to be truly fire-and-forget.	Do not set XA or Exactly-Once unless the reliability level required is once and only once and its possible to use the setting (it is not possible if the client is a HTTP client). If OSB initiates a transaction, it is possible to replace XA with LLR to achieve the same level of reliability.
Disable or delete all log actions.	Log actions add an I/O overhead. Logging also involves an XQuery evaluation which can be expensive. Writing to a single device (resource or directory) can also result in lock contentions.	Disable or delete all log actions.

19.3.3 Tuning XQuery

OSB uses XQuery and XPath extensively for various actions like Assign, Replace, and Routing Table. The following XML structure (\$body) is used to explain XQuery and XPath tuning concepts:

```
<soap-env:Body>
<Order>
<CtrlArea>
<CustName>Mary</CustName>
</CtrlArea>
<ItemList>
<Item name="ACE_Car" >20000 </Item>
<Item name=" Ext_Warranty" >1500</Item>
... a large number of items
</ItemList>
<Summary>
<Total>70000</Total>
<Status>Shipped</Status>
<Shipping>My Shipping Firm </Shipping>
</Summary>
</Order>
</soap-env:Body>
```

You can use the tuning strategies listed in [Table 19–4](#) to tune XQuery.

Table 19–4 XQuery Tuning Strategies

Strategy	Description	Recommendation
Avoid the use of double front slashes ("//") in XPath.	"//" implies all occurrences of a node irrespective of the location in an XML tree. Thus, the entire depth and breadth of the XML tree has to be searched for the pattern specified after a "//".	Use "//" only if the exact location of a node is not known at design time.
Index XPath when applicable.	Indexing helps your system process only what is needed. When indexing, only the top part of the document will be processed by the XQuery engine.	<p>Index an XPath by adding "[1]" after each node of the path.</p> <p>For example, the XPath <code>\$body/Order/CtrlArea/CustName</code> implies returning all instances of <code>Order</code> under <code>\$body</code> and all instances of <code>CtrlArea</code> under <code>Order</code>. The entire document has to be read to correctly process the above XPath.</p> <p>But if you know that there is a single instance of <code>Order</code> under <code>\$body</code> and a single instance of <code>CtrlArea</code> under <code>Order</code>, we can index the above XPath by rewriting it as <code>\$body/Order[1]/CtrlArea[1]/CustName[1]</code>. This will only return the first instances of the child nodes.</p> <p>Note: Do not index when you need a whole array of nodes returned. Indexing will only return the first item node of the array.</p>
Extract frequently used parts of a large XML document as intermediate variables within a FLWOR expression	An intermediate variable can be used to store the common context for multiple values.	Using intermediate variables consumes more memory but reduces redundant XPath processing.
Use a Hybrid Approach for read-only scenarios with Streaming	The gains from streaming can be negated if the payload is accessed a large number of times for reading multiple fields. If all fields read are located in a single subsection of the XML document, a hybrid approach provides the best performance.	<p>Enable streaming at the proxy level and Assigning the relevant subsection to a context variable. The individual fields can then be accessed from this context variable.</p> <p>The fields "Total" and "Status" can be retrieved using three Assign actions:</p> <pre>Assign "\$body/Order[1]/Summary[1]" to "foo" Assign "\$foo/Total" to "total" Assign "\$foo/Status" to "total"</pre>

Note: Pipelines enabled for content streaming should use "XQuery 1.0". Using "XQuery 2004" does work, but incurs significant performance overhead, as there are "on-the-fly" conversions that happen to and from XQuery 1.0 engine. There is a design-time warning to that effect.

19.3.4 Tuning Poller-based Transports

Latency and throughput of poller-based transports depends on the frequency with which a source is polled and the number of files and messages read per polling sweep.

The following are the main transport configurations to tune:

- [Polling Interval](#)
- [Read Limit](#)

19.3.4.1 Polling Interval

Consider using a smaller polling interval for high throughput scenarios where the message size is not very large and the CPU is not saturated. The primary polling interval defaults are listed below with links to additional information:

Polling Intervals	Default Interval	Additional Information
File Transport	60 seconds	"File Transport Configuration Page" in the <i>Developing Services with Oracle Service Bus</i>
FTP Transports	60 seconds	"FTP Transport Configuration Page" in the <i>Developing Services with Oracle Service Bus</i>
MQ Transport	1000 milliseconds	"MQ Transport Configuration Page" in the <i>Developing Services with Oracle Service Bus</i>
SFTP Transport	60 seconds	"SFTP Transport Configuration Page" in the <i>Developing Services with Oracle Service Bus</i>
JCA Transport	60 seconds	"JCA Transport Configuration Page" in the <i>Developing Services with Oracle Service Bus</i>

19.3.4.2 Read Limit

The read limit determines the number of files or messages that are read per polling sweep. You can tune it with the information in [Table 19-5](#).

For more information, see "Using the File Transport" in *Developing Services with Oracle Service Bus*.

Table 19-5 Essential Read Limit Tuning

Parameter	Symptoms if not properly tuned	Tuning Recommendation	Performance Trade-offs
Read Limit Default: 10 for File and FTP transports	Excessive memory use or high memory use due to a large number of files read into memory simultaneously.	Set this value to the desired concurrency. It can be set to 0 to specify no limit. The read limit determines the number of files or messages that are read per polling sweep.	Setting the Read Limit to a high value and the Polling Interval to a small value may result in a large number of messages being simultaneously read into memory. This can lead to an out-of-memory error if the message size is large.

Tuning Oracle Enterprise Scheduler Service

This chapter will describe how to tune Oracle Enterprise Scheduler Service (ESS).

- [Section 20.1, "About Enterprise Scheduler Service"](#)
- [Section 20.2, "Tuning Enterprise Scheduler Service Parameters"](#)

20.1 About Enterprise Scheduler Service

Oracle Enterprise Scheduler enables scheduling and running jobs within a particular time frame, or workshift, using rules to create work assignments. Oracle Enterprise Manager Fusion Applications Control allows you to define, control and manage Oracle Enterprise Scheduler job metadata, including job definitions, job requests, job sets (a collection of job requests), incompatibilities (job definitions and job sets that cannot run at the same time for a given application) and schedules governing the execution of job requests.

For more information, see "Introduction to Administering Oracle Enterprise Scheduler" in *Administering Oracle Enterprise Scheduler*.

20.2 Tuning Enterprise Scheduler Service Parameters

There are two knobs that can be tuned, and their strategies are described in [Table 20–1](#).

Table 20–1 Essential Enterprise Scheduler Service Tuning

Name	Symptoms	Recommendations	Trade-offs
Maximum Poll Interval Default: 15 seconds	A high number of requests whose execution time has been reached and remain in WAIT state for an extended time.	Decrease this value if there is an excess of waiting requests that are eligible to be dispatched and processor threads are available.	Lowering the value will increase CPU usage and database activity. Increasing the value may delay the dispatching of requests that are ready for processing.
Thread Count Default: 25	A high number of requests in READY state that are otherwise available for processing.	Increase this value if there is a build up of requests that are ready to be executed and the increase system resource usage is acceptable. Lower the value to reduce the amount of system resources used for request processing.	Increasing this value will increase CPU usage, memory usage, and database activity. Lowering this value may result in a build up and potentially delay processing of requests.

