# Oracle® Fusion Middleware

Performance and Tuning Guide

11g Release 2 (11.1.2.2.0)

**E28552-06**

June 2014

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

ORACLE

Oracle Fusion Middleware Performance and Tuning Guide , 11g Release 2 (11.1.2.2.0)

E28552-06



Primary Author: Lisa Jamen

# Contents

## 6 Oracle HTTP Server Performance Tuning

## 9 Oracle TopLink (EclipseLink) JPA Performance Tuning



## 10 Oracle Web Cache Performance Tuning



# Part IV SOA Suite Components

## 11 General Tuning for SOA Suite Components

## 15 Oracle Mediator Performance Tuning



## 16 Oracle Business Process Management Performance Tuning



## 17 Oracle Human Workflow Performance Tuning

## 24 Oracle Virtual Directory Performance Tuning

## 25 Oracle Access Management Performance Tuning

## 26 Oracle Identity Manager Performance Tuning



## 27 Oracle Adaptive Access Manager Performance Tuning

## Part VI Oracle WebCenter Components

## 31 Oracle WebCenter Portal Performance Tuning



## Part VII Capacity Planning, Scalability, and Availability

## 32 Capacity Planning



## 33 Using Clusters and High Availability Features

# 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

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

## 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`.

### Accessibility of Code Examples in Documentation

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

### Accessibility of Links to External Web Sites in Documentation

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

### Access to Oracle Support

Oracle customers 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. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

# Part I

# Introduction

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

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

# 1 Introduction and Roadmap

This section describes the contents and organization of this guide.



## 1.1 Document Scope and Audience

*Oracle Fusion Middleware Performance and Tuning Guide* 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

- 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, "Understanding the Oracle Dynamic Monitoring Service" provides an overview and features available in the Oracle Dynamic Monitoring Service (DMS).
- Chapter 6, "Oracle HTTP Server Performance Tuning," discusses the techniques for optimizing Oracle HTTP Server performance, the Web server component for Oracle Fusion Middleware. It provides a listener for Oracle WebLogic Server and the framework for hosting static pages, dynamic pages, and applications over the Web.
- Chapter 7, "Oracle Metadata Service (MDS) Performance Tuning," provides tuning tips for Oracle Metadata Service (MDS). MDS is used by components such as

Oracle WebCenter Framework and Oracle Application Development Framework to manage metadata.

- Chapter 8, "Oracle Application Development Framework Performance Tuning," provides basic guidelines on how to maximize the performance and scalability of the ADF stack in applications. Oracle ADF is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications. This chapter covers design time, configuration time, and deployment time performance considerations.
- Chapter 9, "Oracle TopLink (EclipseLink) JPA Performance Tuning," provides some of the available performance options for Java Persistence API (JPA) entity architecture. Oracle TopLink includes EclipseLink as the JPA implementation.
- Chapter 10, "Oracle Web Cache Performance Tuning," provides methods and guidelines for improving the performance of Oracle Application Server Web Cache (Oracle Web Cache). Oracle Web Cache is a content-aware server accelerator or reverse proxy that improves the performance, scalability, and availability of Web sites that run on Oracle Fusion Middleware.
- Chapter 11, "General Tuning for SOA Suite Components," 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, "Oracle Business Rules Performance Tuning" 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 13, "Oracle BPEL Process Manager Performance Tuning," provides several BPEL property settings that can be configured to optimize performance at the process, domain, and application server levels. This chapter describes these property settings and provides recommendations on how to use them.
- Chapter 14, "Oracle Business Activity Monitoring Performance Tuning" 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.
- Chapter 15, "Oracle Mediator Performance Tuning," describes how to tune Oracle Mediator, a service engine within the Oracle SOA Service Infrastructure, for optimal performance. Oracle Mediator provides the framework to mediate between various providers and consumers of services and events. The Mediator service engine runs with the SOA Service Infrastructure Java EE application.
- Chapter 16, "Oracle Business Process Management Performance Tuning" describes how to tune Oracle Service Bus (OSB) which provides connectivity, routing, mediation, management and also some process orchestration capabilities.
- Chapter 17, "Oracle Human Workflow Performance Tuning," describes how to tune Oracle Human Workflow for optimal performance. Oracle Human Workflow is a service engine running in Oracle SOA Service Infrastructure that allows the execution of interactive human driven processes. A human workflow provides the human interaction support such as approve, reject, and reassign actions within a process or outside of any process. The Human Workflow service consists of a

number of services that handle various aspects of human interaction with a business process.

- Chapter 18, "Oracle Adapters Performance Tuning," describes how to tune Oracle Adapters for optimal performance. Oracle technology adapters integrate Oracle Application Server and Oracle Fusion Middleware components such as Oracle BPEL Process Manager (Oracle BPEL PM) or Oracle Mediator components to file systems, FTP servers, database queues (advanced queues, or AQ), Java Message Services (JMS), database tables, and message queues (MQ Series).
- Chapter 19, "User Messaging Service Performance Tuning," describes tips for tuning the User Messaging Service. Oracle User Messaging Service (Oracle UMS) enables two way communications between users and deployed applications. It has support for a variety of channels, such as E-mail, IM, SMS, and text-to-voice messages. Oracle UMS is integrated with Oracle Fusion Middleware components, such as Oracle BPEL PM, Oracle Human Workflow, Oracle BAM and Oracle WebCenter.
- Chapter 20, "Oracle B2B Performance Tuning" provides tuning tips for Oracle B2B. Oracle B2B is an e-commerce gateway that enables the secure and reliable exchange of business documents between an enterprise and its trading partners. Oracle B2B supports business-to-business document standards, security, transports, messaging services, and trading partner management. With Oracle B2B used as a binding component within an Oracle SOA Suite composite application, end-to-end business processes can be implemented.
- Chapter 21, "Oracle Service Bus Performance Tuning," provides basic and advanced tuning tips and design considerations for Oracle Service Bus.
- Chapter 22, "Oracle Business Intelligence Performance Tuning," provides basic and advanced tuning tips for Oracle Business Intelligence.
- Chapter 23, "Oracle Internet Directory Performance Tuning," provides guidelines on Oracle Internet Directory tuning and configuration requirements. Oracle Internet Directory is an LDAP Version 3-enabled service that enables fast retrieval and centralized management of information about dispersed users, network configuration, and other resources.
- Chapter 24, "Oracle Virtual Directory Performance Tuning," provides tuning tips for Oracle Virtual Directory. Oracle Virtual Directory is an LDAP Version 3-enabled service that provides an abstracted view of one or more enterprise data sources. Oracle Virtual Directory consolidates multiple data sources into a single directory view, enabling you to integrate LDAP-aware applications with diverse directory server data stores.
- Chapter 25, "Oracle Access Management Performance Tuning," describes tuning Oracle Access Management which includes a full range of services that provide Web perimeter security functions and Web single sign-on; identity context, authentication and authorization; policy administration; testing; logging; auditing; and more.
- Chapter 26, "Oracle Identity Manager Performance Tuning," describes tuning Oracle Identity Manager (OIM) which provides operational and business efficiency through centralized administration and complete automation of identity and user provisioning events across the enterprise, as well as extranet applications.
- Chapter 27, "Oracle Adaptive Access Manager Performance Tuning," describes tuning Oracle Adaptive Access Manager (OAAM) which provides real-time or batch risk analysis and adaptive authentication capabilities to actively prevent

fraud. Out of the box integrations with Oracle Identity Manager (OIM) and Oracle Access Manager (OAM) secure web single sign-on and self-service password management flows with adaptive authentication.

- Chapter 28, "Oracle Unified Directory Performance Tuning," describes tuning Oracle Unified Directory which is a comprehensive next generation directory service that is designed to address large deployments, to provide high performance, to be highly extensive and to be easy to deploy, manage, and monitor.
- Chapter 29, "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 30, "Oracle Entitlements Server Performance Tuning" provides guidelines for tuning and sizing Oracle Entitlements Server (OES).
- Chapter 31, "Oracle WebCenter Portal Performance Tuning," provides suggested tuning tips for Oracle WebCenter including: Environment Configuration, Application Configuration and Back-End Services and Server Configuration.
- Chapter 32, "Capacity Planning," discusses the process of determining what type of hardware and software configuration is required to meet application needs.
- Chapter 33, "Using Clusters and High Availability Features," discusses the architecture, interaction, and dependencies of Oracle Fusion Middleware components, and explains how they can be deployed in a high availability architecture to maximize performance.

## 1.3 How To Use the Performance Recommendations in this Guide

This guide provides detailed examples of specific tuning parameters and metrics for the Oracle Fusion Middleware components. Be sure that you understand your performance requirements and deployment topologies before implementing any changes. **You should always consult your own use cases before implementing any of the recommended configurations in this guide.**

There are different levels of tuning configurations included this guide:

- Review Chapter 2, "Top Performance Areas" to address common performance bottlenecks, hardware resource issues and basic application tuning *before* you begin any component-specific tuning.
- Each component chapter includes **Basic Tuning Configurations** that should apply to most component deployments.
- **Advanced Tuning Configurations** are also provided, but may not apply to all deployments.
- Some component chapters include **Specific Use Case Tuning Configurations** that describe tuning recommendations for specific usage scenarios and tasks.

## 1.4 Related Documentation

For more information, see the following documents in the Oracle Fusion Middleware 11g Release 2 (11.1.2) documentation set:

- *Oracle Fusion Middleware Administrator's Guide*

- *Oracle Fusion Middleware 2 Day Administration Guide*
- *Oracle Fusion Middleware Concepts*
- *Oracle Fusion Middleware Application Security Guide*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*
- *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*
- *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*
- *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*
- *Oracle Unified Directory Administrator's Guide*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*

# 2

# Top Performance Areas

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



## 2.1 About Identifying Top Performance Areas

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

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

*Table 2–1 Top Performance Areas for Oracle Fusion Middleware*

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

## 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 "System Requirements and Prerequisites" in the *Oracle Fusion Middleware Installation Planning Guide* for your platform.

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

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

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

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

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

Oracle Fusion Middleware provides a variety of mechanisms to help you control resource concurrency; this can limit the impact of bursts of traffic. However, for a consistently saturated system, these mechanisms should be viewed as temporary solutions. For more information see Section 2.9, "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. This section discusses the tuning options that have the greatest impact on performance.

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

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

---

**Note:** For additional information about tuning the JVM, see the following:

- Java Performance Documentation: (http://java.sun.com/docs/performance/)
- The Java Tuning White Paper (http://java.sun.com/performance/reference/whitepapers/tuning.html)
- Garbage Collection Tuning (http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html)
- "Tuning Java Virtual Machines (JVMs)" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*

---

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

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

## 2.4.1 Configuring Garbage Collection

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

This section cover the following Garbage Collection tuning options:

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

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

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

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

1. How often is garbage collection taking place? Compare the time stamps around the garbage collection.
2. How long is a full garbage collection taking?

3. What is the heap size after each full garbage collection? If the heap is always 85 percent free, for example, you might set the heap size smaller.
4. Do the young generation heap sizes (Sun) or Nursery size (Jrockit) need tuning?

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

- Sun JVM command line options:

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

  Look for "Full GC" to identify major collections.

- Additional Sun Tools:
  - JStat
  - JConsole
  - Visualgc

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

- Jrockit JVM command line options:

  ```
  -XXverbose:gc
  ```

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

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

#### 2.4.1.1 Specifying Heap Size Values

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

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

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

In production environments, set the minimum heap size and the maximum heap size to the same value to prevent wasting virtual machine resources used to constantly grow and shrink the heap. Ensure that the sum of the maximum heap size of all the JVMs running on your system does not exceed the amount of available physical RAM. If this value is exceeded, the Operating System starts paging and performance degrades significantly. The virtual machine always uses more memory than the heap size. The memory required for internal virtual machine functionality, native libraries outside of the virtual machine, and permanent generation memory (memory required to store classes and methods) is allocated in addition to the heap size settings.

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

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

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

---

**See Also:** For more information on how to specify heap size values for Oracle WebLogic Server, see "Specifying Heap Size Values" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*

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

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

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

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

---

### 2.4.1.2 Selecting a Garbage Collection Scheme

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

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

- For an overview of the garbage collection schemes available with Sun's HotSpot VM, see "Java SE 6 HotSpot Virtual Machine Garbage Collection Tuning" at

http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html.

- For a comprehensive explanation of the collection schemes available, see "Memory Management in the Java HotSpot™ Virtual Machine" at http://java.sun.com/j2se/reference/whitepapers/memorymanagement_whitepaper.pdf.
- For a discussion of the garbage collection schemes available with the JRockit JDK, see "Using the JRockit Memory Management System" at http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html.

#### 2.4.1.3 Disabling Explicit Garbage Collection

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

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

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

- For Oracle JRockit virtual machines use `-XXnoSystemGC`

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

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

### 2.4.2 Logging Low Memory Conditions

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

---

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

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

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

---

### 2.4.3 Monitoring and Profiling the JVM

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

- **Oracle JRockit® Mission Control**

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

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

- **Sun JVM**

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

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

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

## 2.5 Tuning the WebLogic Server

If your Oracle Fusion Middleware applications are using the WebLogic Server, see "Tuning Java Virtual Machines (JVMs)" in *Oracle Fusion Middleware Performance and Tuning for 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.

This section covers the following:

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

---

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

---

## 2.6.1 Tuning Database Parameters

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

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

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

### 2.6.1.1 Initialization Parameters for Oracle 10g

The following table describes several performance-related database initialization parameters for Oracle 10*g* database. The tuning considerations listed below are applicable to most scenarios. Always set your database parameters based on your own use case scenarios.

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

| Database Parameter | Description |
|---|---|
| _b_tree_bitmap_plans | Consider setting this parameter to FALSE to prevent optimizer from attempting bitmap operations as there are no bitmap indexes in Fusion Middleware. |
| DB_BLOCK_SIZE | DB_BLOCK_SIZE specifies (in bytes) the size of Oracle database blocks. The default block size of 8K is optimal for most systems. Set this parameter at the time of database creation. |
| NLS_SORT | Consider setting NLS_SORT to BINARY, otherwise sort will do full table scan and performance can be impacted. |
| OPEN_CURSORS | Consider using a value of 500 open cursors (handles to private SQL areas) a session can have at once. |
| SESSION_CACHED_CURSORS | Consider using a value of 500 session cursors to cache. |
| SESSION_MAX_OPEN_FILES | SESSION_MAX_OPEN_FILES specifies the maximum number of BFILEs that can be opened in any session. Once this number is reached, subsequent attempts to open more files in the session by using DBMS_LOB.FILEOPEN() or OCILobFileOpen() may fail. The maximum value for this parameter depends on the equivalent parameter defined for the underlying operating system. |
| JOB_QUEUE_PROCESSES | JOB_QUEUE_PROCESSES specifies the maximum number of processes that can be created for the execution of jobs. It specifies the number of job queue processes per instance. |
| LOG_BUFFER | LOG_BUFFER specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. The LGWR process writes redo log entries from the log buffer to a redo log file. |
| UNDO_MANAGEMENT | UNDO_MANAGEMENT specifies which undo space management mode the system should use. When set to AUTO, the instance starts in automatic undo management mode. In manual undo management mode, undo space is allocated externally as rollback segments. |
| PL_SQL_CODE_TYPE | Consider setting PL_SQL_CODE_TYPE to NATIVE |
| PROCESSES | Consider using a value of 5000 operating system processes to be connected to Oracle concurrently. |
| PGA_AGGREGATE_TARGET | Consider setting PGA_AGGREGATE_TARGET to 1G of PGA memory available to all server processes attached to the instance. |

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

| Database Parameter | Description |
|---|---|
| SGA_MAX_SIZE | Consider setting the SGA_MAX_SIZE to 2G initially and then monitor the production database on daily basis and adjust SGA and PGA accordingly. |
| SGA_TARGET | Consider setting the SGA_TARGET to 2G initially and then monitor the production database on daily basis and adjust SGA and PGA accordingly. |
| TRACE_ENABLED | TRACE_ENABLED controls tracing of the execution history, or code path, of Oracle. Oracle Support Services uses this information for debugging.<br>Although the performance impacts incurred from processing is not excessive, you may improve performance by setting TRACE_ENABLED to FALSE. |

#### 2.6.1.2 Initialization Parameters for Oracle 11g

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

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

| Database Parameter | Description |
|---|---|
| AUDIT_TRAIL | If there is NO policy to audit db activity, consider setting this parameter to NONE. Enabling auditing can impact performance. |
| MEMORY_MAX_TARGET | MEMORY_MAX_TARGET specifies the maximum value to which a DBA can set the MEMORY_TARGET initialization parameter. |
| MEMORY_TARGET | Consider setting the MEMORY_TARGET to NONE. Set SGA and PGA separately as setting MEMORY_TARGET does not allocate sufficient memory to SGA and PGA as needed. |
| PGA_AGGREGATE_TARGET | Consider using a value of 1G for PGA initially and then monitor the production database on daily basis and adjust SGA and PGA accordingly.<br>If the database server has more memory, consider setting PGA_AGGREGATE_TARGET to a value higher than 1G based on usage needs. |
| SGA_MAX_SIZE | Consider setting MEMORY_TARGET instead of setting SGA and the PGA separately. |
| SGA_TARGET | Consider using a value of 2G for SGA is 2G to start with and initially and then monitor the production database on daily basis and adjust SGA and PGA accordingly.<br>If the database server has more memory, consider setting SGA_TARGET to a value higher than 2G based on usage needs. |

### 2.6.2 Tuning Redo Logs Location and Sizing

Tuning the redo log options can provide performance improvement for applications running in an Oracle Fusion Middleware environment, and in some cases, you can significantly improve I/O throughput by moving the redo logs to a separate disk.

To manage REDO logs and UNDO change logs as a DB user with `sysdba` role, do the following:

- Increase allocated disk space for redo log files. It is recommended to have three redo log files. Each file must be at least 4GB in size.
- Increase disk space allocated for UNDO table space. It is recommended to have a 20GB UNDO tablespace.

For more information about Oracle Database Redo Log, see "Managing the Redo Log" in the *Oracle Database Administrator's Guide*.

For more information on general database tuning, see the *Oracle Database Performance Tuning Guide*.

### 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`.

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

## 2.7 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 *Oracle Fusion Middleware Configuring and Managing 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.

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

- Statement Cache Type—The algorithm that determines which statements to store in the statement cache.

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

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

For more information on using statement caching, see "Increasing Performance with the Statement Cache" in the *Oracle Fusion Middleware Configuring and Managing JDBC 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 Sever Thread Pool
- Tuning Oracle WebCenter Concurrency
- Tuning BPEL Concurrency

### 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 *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

You can use the `MaxClients` and `ThreadsPerChild` directives to limit incoming requests to WebLogic instances from the Oracle HTTP Server based on your expected client load and system resources. The following sections describe some Oracle HTTP Server tuning parameters related to connection limits that you typically need to tune based on your expected client load. See Chapter 6, "Oracle HTTP Server Performance Tuning" for more information and a more complete list of tunable parameters.

#### 2.9.1.1 MaxClients/ThreadsPerChild

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

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

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

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

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

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

---

**Note:** ThreadsPerChild, StartServers, and ServerLimit properties are inter-related with the MaxClients setting. All of these properties must be set appropriately to achieve the number of connections as specified by MaxClients. See Table 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 *Oracle Fusion Middleware Administrator's Guide for 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 Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

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

---

### 2.9.3 Tuning the WebLogic Sever Thread Pool

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

The common thread pool changes its size automatically to maximize throughput. WebLogic Server monitors throughput over time and based on history, determines whether to adjust the thread count. For example, if historical throughput statistics

indicate that a higher thread count increased throughput, WebLogic increases the thread count. Similarly, if statistics indicate that fewer threads did not reduce throughput, WebLogic decreases the thread count.

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

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

- The default fair share is not sufficient.

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

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

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

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

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

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

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

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

### 2.9.4 Tuning Oracle WebCenter Concurrency

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

### 2.9.5 Tuning BPEL Concurrency

The Oracle BPEL Process Manager has its own thread controls and specialized tuning. See Section 13.2.1, "BPEL Threading Model".

## 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.

# 3 Performance Planning

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

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

## 3.1 About Oracle Fusion Middleware Performance Planning

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

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

## 3.2 Performance Planning Methodology

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

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

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

### 3.2.1 Define Your Performance Objectives

Before you can begin performance tuning your applications, you must first identify the performance objectives you hope to achieve. To determine your performance

objectives, you must understand the applications deployed and the environmental constraints placed on the system.

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

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

Performance objectives are limited by constraints, such as:

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

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

  Capacity planning requires assessment of your system performance goals and an understanding of your application. Capacity planning for server hardware should focus on maximum performance requirements. For more information on capacity planning, see Chapter 32, "Capacity Planning".
- The configuration of high availability architecture to address peak usage and response times. For more information on implementing high availability features in Oracle Fusion Middleware applications, see Chapter 33, "Using Clusters and High Availability Features".
- The ability to interoperate between domains, use legacy systems, support legacy data.
- Development, implementation, and maintenance costs.

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

#### 3.2.1.1 Define Operational Requirements

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

- Application Architecture
- Security Requirements
- Hardware Resources

#### 3.2.1.2 Identify Performance Goals

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

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

- Anticipated number of users

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

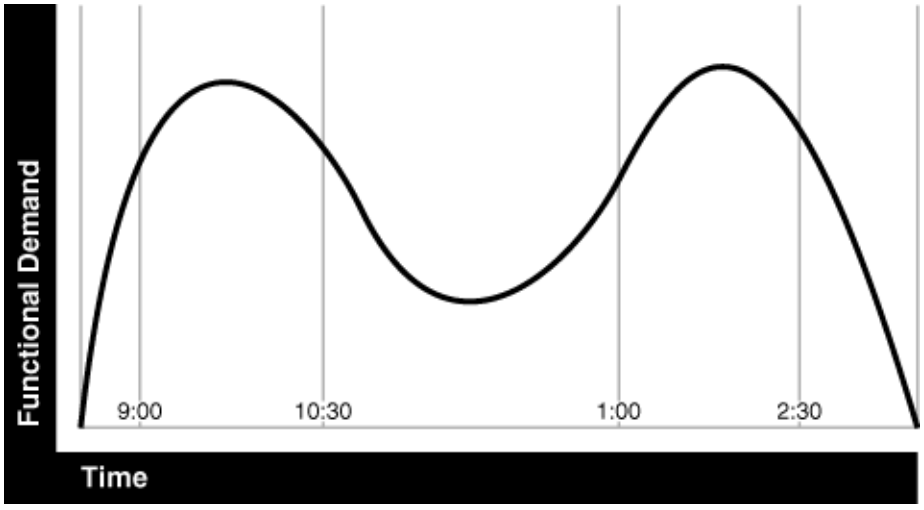
#### 3.2.1.3 Understand User Expectations

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

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

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

***Figure 3–1 Adjusting Capacity and Functional Demand***




#### 3.2.1.4 Conduct Performance Evaluations

With clearly defined performance goals and performance expectations, you can readily determine when performance tuning has been successful. Success depends on the functional objectives you have established with the user community, your ability to measure whether the criteria are being met, and your ability to take corrective action to overcome any exceptions.

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

### 3.2.2 Design Applications for Performance and Scalability

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

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

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

### 3.2.3 Monitor and Measure Your Performance Metrics

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

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

---

**See Also:** *Oracle Database Performance Tuning Guide*

*Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*

*Oracle Fusion Middleware Administrator's Guide*

---

# 4

# Monitoring Oracle Fusion Middleware

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

This chapter contains the following sections:



> **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.

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

- Oracle Enterprise Manager Fusion Middleware Control. See Section 4.2.
- Oracle WebLogic Server Administration Console. See Section 4.3.
- Oracle WebLogic Diagnostics Framework (WLDF). See Section 4.4.
- Oracle WebLogic Scripting Tool (WLST). See Section 4.5.
- DMS Spy Servlet. See Section 4.6.
- Oracle Process Manager and Notification Server. See Section 4.7.

- Oracle Enterprise Manager Cloud Control. See Section 4.8.
- Operating System Performance Commands. See Section 4.9.
- Network Performance Monitoring Tools. See Section 4.10.

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

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

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

### 4.1.1 Measuring Your Performance Metrics

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

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

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

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

## 4.2 Oracle Enterprise Manager 11g Fusion Middleware Control

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

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

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

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

### 4.2.1 Viewing Performance Metrics Using Fusion Middleware Control

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

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

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

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

Oracle Fusion Middleware automatically and continuously measures run-time performance. The performance metrics are automatically enabled; you do not need to set options or perform any extra configuration to collect them. If you are interested in viewing historical data, consider using Oracle Enterprise Manager Cloud Control. For more information, see Section 4.8, "Oracle Enterprise Manager Cloud Control".

## 4.3 Oracle WebLogic Server Administration Console

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

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

Use the WebLogic Server Administration Console to:

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

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

**Additional WebLogic Server Console Resources:**

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

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

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

## 4.4 WebLogic Diagnostics Framework (WLDF)

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

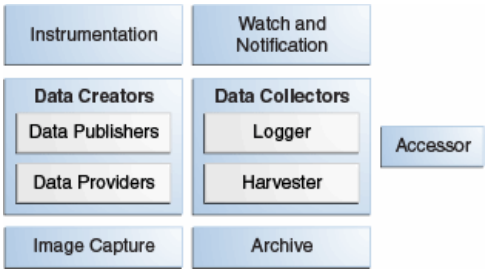
WLDF includes several components for collecting and analyzing data:

- **Data Creators**— data publishers and data providers that are distributed across WLDF components.
- **Diagnostic Image Capture**—Creates a diagnostic snapshot from the server that can be used for post-failure analysis.
- **Archive**—Captures and persists data events, log records, and metrics from server instances and applications.

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

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

***Figure 4–1 Major WLDF Components***




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

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

## 4.5 WebLogic Scripting Tool (WLST)

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

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

- Interactively, on the command line

- In script mode, supplied in a file
- Embedded in Java code

**See Also:**

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

### 4.5.1 Using Custom WLST Commands

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

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

#### 4.5.1.1 Using WLST Commands for System Components

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

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

**Note:** The `dmstool` command has been replaced with the following commands: `dumpMetrics`, `displayMetricTables`, `displayMetricTableNames`.

## 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`: `<ORACLE_HOME>/modules/oracle.dms_11.1.1/dms.war`.

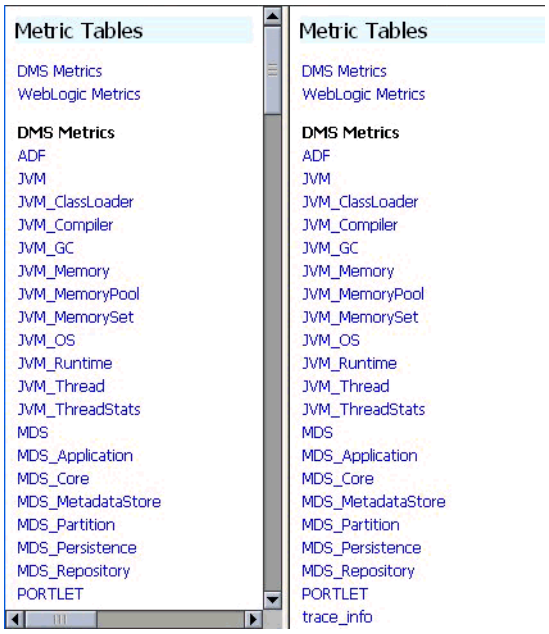
The DMS web application is deployed by default as part of a JRF-enabled server instance. The URL is: `http://host:port/dms/Spy`.

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

### 4.6.2 Using the DMS Spy Servlet

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

***Figure 4–2 Spy Servlet Page - Metrics Tables***

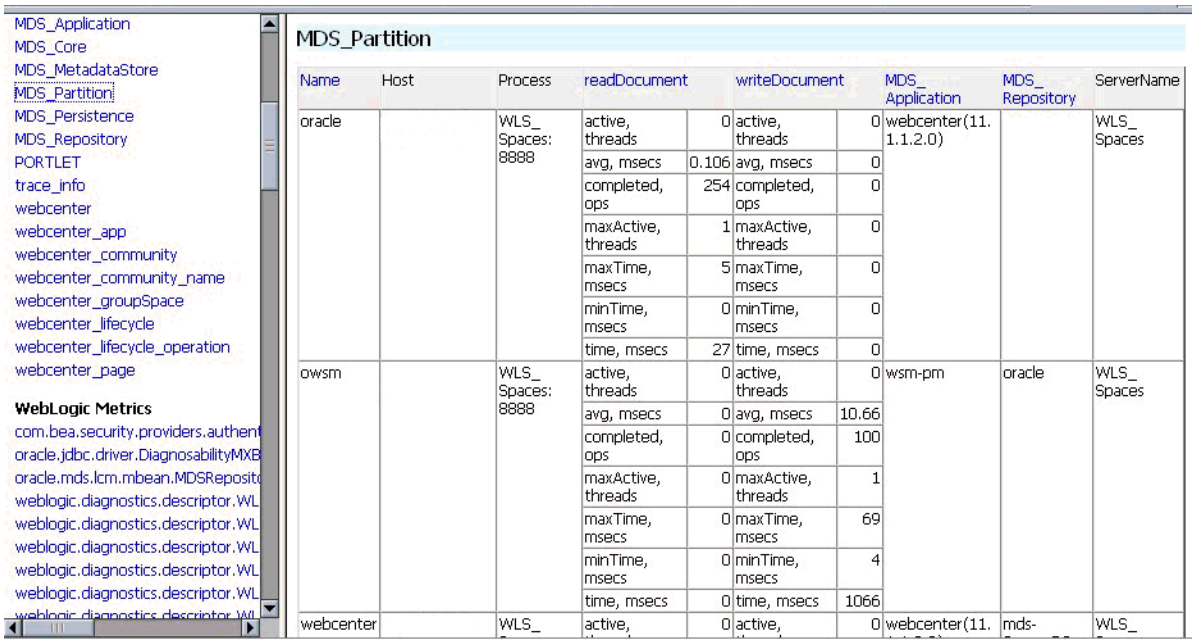



Note that the Spy servlet can display metric tables for WebLogic Server and also for non-Java EE components that are deployed.

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 4–3 shows the MDS_Partition table.

***Figure 4–3 MDS Partition Table***




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

## 4.7 Oracle Process Manager and Notification Server

Oracle Process Manager and Notification Server (OPMN) monitors the status of Oracle Fusion Middleware components. You can also start and stop system components, monitor system components, and perform many other tasks related to process management. For example, you can use OPMN to start and stop OPMN-managed processes, such as Oracle HTTP Server and Oracle Web Cache. For more information on OPMN commands, see "Section 6.2, "Monitoring Oracle HTTP Server Performance".

---

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

---

## 4.8 Oracle Enterprise Manager Cloud Control

Oracle Enterprise Manager is Oracle's integrated enterprise information technology (IT) management product line, which provides the industry's only complete, integrated, and business-driven enterprise cloud management solution. Oracle Enterprise Manager creates business value for IT by leveraging the built-in management capabilities of the Oracle stack for traditional and cloud environments, enabling customers to achieve unprecedented efficiency gains while dramatically increasing service levels.

The key capabilities of Enterprise Manager include:

- A complete cloud lifecycle management solution enabling you to quickly set up, manage, and support enterprise clouds and traditional Oracle IT environments from applications to disk.
- Maximum return on IT management investment through the best solutions for intelligent management of the Oracle stack and engineered systems with real-time integration of Oracle's knowledge base with each customer environment.
- Best service levels for traditional and cloud applications through business-driven application management.

For more information about the Oracle Enterprise Manager Cloud Control, refer to *Oracle Enterprise Manager Cloud Control Introduction*.

## 4.9 Native Operating System Performance Commands

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

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

## 4.10 Network Performance Monitoring Tools

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

# Part II

## Core Components

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

- Chapter 5, "Understanding the Oracle Dynamic Monitoring Service"
- Chapter 6, "Oracle HTTP Server Performance Tuning"
- Chapter 7, "Oracle Metadata Service (MDS) Performance Tuning"

---

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

---

# 5

# Understanding the Oracle Dynamic Monitoring Service

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



## 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. Specifically, DMS is used by Oracle WebCache, Oracle HTTP Server (OHS), Oracle Application Development Framework (ADF), WebLogic Diagnostic Framework (WLDF), and JDBC. 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 Tracing and Events
- DMS Nouns
- DMS Sensors

#### 5.1.1.1 DMS Tracing and Events

Table 5–1 provides a list of DMS tracing and event terminology.

*Table 5–1 DMS Tracing and Event Terminology*

| DMS Term | Definition |
|---|---|
| Condition | 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.<br>Two types of condition exist:<br>▪ Noun Type Condition - operates on the name of the noun type associated with a sensor or noun event.<br>▪ Context Condition - operates on the values currently set within the current Execution Context.<br>For more information on using conditions, see Section 5.7, "DMS Tracing and Events". |
| Destination | 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 JRockit Flight Recorder, yet another might render information gleaned from incoming events as data in an MBean. |
| Event Route | An **event route** connects a filter to a destination. Event routes may be enabled or disabled. For event tracing to be activated for a specific filter, one or more event routes must exist for that filter and must be enabled. |
| Filter | 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.<br>For example it is possible to define filters to:<br>▪ Only pass sensor updates that are made when the execution context has a key-value pair of "role"-"admin"<br>▪ Only pass sensor updates from nouns of type "JDBC_ Statement"<br>For more information on using filters, see Section 5.7, "DMS Tracing and Events". |
| Listener | A DMS **listener** is also known as the destination. See Section 5.7.2, "Configuring Destinations" for more information. |

#### 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 a name that reflects the set of metrics being collected.

**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 5–2 shows DMS replacement for special characters in names.

***Table 5–2 Replacement for Special Characters in DMS Names***

| Character | DMS Replacement Character |
|---|---|
| Space character | Underscore character: _ |
| Period character: . | Underscore character: _ |
| Control character | Underscore character: _ |
| Less than character: < | Open parenthesis: ( |
| Greater than character: > | Close parenthesis: ) |
| Ampersand: & | Caret: ^ |
| Double quote: " | Backquote: ' |
| Single quote: ' | Backquote: ' |

---

**Note:** Oracle Fusion Middleware includes several built-in metrics. The Oracle Fusion Middleware built-in metrics do not always follow the DMS naming conventions.

---

#### 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 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 5.1.1.3.1, "DMS PhaseEvent Sensors"
- Section 5.1.1.3.2, "DMS Event Sensors"
- Section 5.1.1.3.3, "DMS State Sensors"

**5.1.1.3.1 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 5–3 lists the metrics available with PhaseEvent sensors.

***Table 5–3 DMS PhaseEvent Sensor Metrics***

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

**5.1.1.3.2 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 5–4 describes the metric that is associated with an event sensor.

***Table 5–4 DMS Event Sensor Metrics***

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

**5.1.1.3.3 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 5–5 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 5–5 DMS State Sensor Metrics***

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

**5.1.1.3.4 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 5–3, Table 5–4, and Table 5–5.

## 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, see the Oracle Fusion Middleware Certification Matrix at http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.

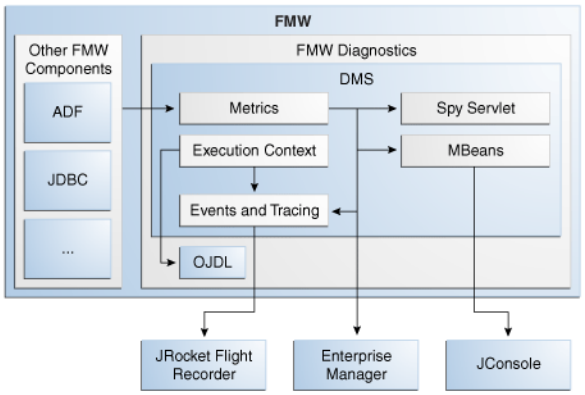
## 5.3 Understanding DMS Architecture

DMS consists of the following features:

- **DMS Metrics** - The DMS metrics feature provides Java and C APIs that are also used by other Oracle Fusion Middleware components for instrumenting code with performance measurements and other useful state metrics. In addition, the metrics feature provides an aggregation language for computing derived metrics and tools for accessing the metrics.
- **Execution Context** - Execution Context supports the maintenance and propagation of a specific context structure throughout the Oracle stack. By making the context structure available consistently across all Oracle code, the potential for cross component and cross product correlation of diagnostic data increases. 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 Section 4.2.1, "Viewing Performance Metrics Using Fusion Middleware Control" 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 HTTP servers and Oracle WebCache.

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
- Viewing metrics using WSADMIN (IBM WebSphere)

### 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 7001.

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_11.1.1/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 *Oracle Fusion Middleware 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... |
|---|---|
| displayMetricTableNames | List the names of the available metric tables. |
| displayMetricTables | Show the content of the DMS metric tables. |
| dumpMetrics | Display metrics in the internal format. Valid formats for the dumpMetrics command include raw, xml and pdml. |

As well as displaying textual output, theses 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 Section 4.5.1.1, "Using WLST Commands for System 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 4.2.1, "Viewing Performance Metrics Using Fusion Middleware Control".

### 5.4.6 Viewing metrics using WSADMIN (IBM WebSphere)

The following commands can be used with IBM WebSphere to display the following:

| Use this command... | To do this... |
|---|---|
| OracleDMS.displayMetricTableNames() | List the names of the available metric tables. |
| OracleDMS.displayMetricTables() | Show the content of the DMS metric tables. |
| OracleDMS.dumpMetrics() | Display metrics in the internal format. Valid formats include raw, xml and pdml. |

For more information on using IBM WebSphere, see "Managing Oracle Fusion Middleware on IBM WebSphere" in the *Oracle Fusion Middleware Third-Party Application Server Guide*.

## 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//f
orceBalance_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 *Oracle Fusion Middleware 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 HTTP 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 HTTP request sent directly to Oracle WebLogic Server from a browser:
   - Root task only on Oracle WebLogic Server
2. An HTTP request sent through Oracle HTTP Server (acting as a reverse proxy) to Oracle WebLogic Server:
   - Root task on Oracle HTTP Server
   - Single sub-task on Oracle WebLogic Server
3. An HTTP request sent from Oracle HTTP 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 HTTP 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 HTTP request is sent from Oracle HTTP 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 HTTP 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:

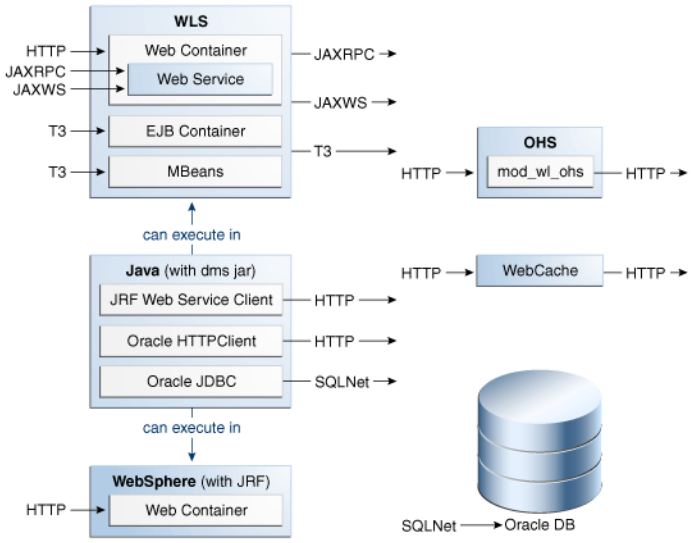
```
displayLogs(ecid="B5C094FA...BE4AE8");
```

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

Starting with Oracle Fusion Middleware 11g Release 1 (11.1.1.3.0), 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)
- HTTP 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 *Oracle Fusion Middleware Administrator's Guide*.

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=JMXEventConfig`

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
- HTTP Request Tracker Destination
- JRockit Flight Recorder Destination

#### 5.7.2.1 LoggerDestination

| | |
|---|---|
| **Description** | The LoggerDestination writes each event to the associated logger. |
| **Implementing Class** | oracle.dms.trace2.runtime.LoggerDestination |
| **Properties** | |
| `loggerName` | 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 *Oracle Fusion Middleware Administrator's Guide*.]

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 DMS Event Output" 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/Middleware/user_projects/domains/base_domain/config/fmwconfig/servers/AdminServer/logging.xml |
| WAS ND | OracleAdminServer | ORACLE_HOME/Middleware/was_profiles/DefaultTopology/DefaultServer/config/cells/DefaultCell/nodes/<nodename>/servers/OracleAdminServer/fmwconfig/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 *Oracle Fusion Middleware 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 HTTP Request Tracker Destination

| | |
|---|---|
| **Description** | The HTTP Request Tracker destinations maintains a list of active HTTP requests, and makes the requests accessible to other Diagnostic Framework (DFW) components. |
| **Implementing Class** | oracle.dms.event.HTTPRequestTrackerDestination |
| **Properties** | |
| `excludeHeaderNames` | Comma separated list of header names to exclude from tracking |

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

For each HTTP request the following information will be dumped:

- URI (such as `/webcenter/home`)
- Start time of the request
- ECID
- Query string
- HTTP Headers

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

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

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

```
> executeDump(name="http.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 JRockit Flight Recorder Destination

The JRockit Flight Recorder (JFR) records information regarding the runtime status and behavior of the JRockit JVM. JFR also exposes an API through which third party events can be reported. JFR is available in JRockit R28 and beyond.

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 | |

*Table 5–6 (Cont.) Values of the JFR Event for a Noun Type*

| Value Name | Description | Relational | Notes |
|---|---|---|---|
| <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 that 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 | JRockit Flight Recorder (JFR) |
|---|---|
| **Noun type:**<br>`JDBC_Connection`<br><br>**Noun path:**<br>`/JDBC/Driver/CONNECTION_7`<br><br>**Sensors:**<br>`CreateStatement (P)`<br>`CreateNewStatement (P)`<br>`DBWaitTime (P)`<br>`JDBC_Connection_Url (S)`<br>`JDBC_Connection_Username (S)`<br><br>**Where:**<br>P: Phase Sensor<br>S : State Sensor<br>E : Event Sensor | Producer Name: JDBC<br>The Producer Name is based on the leading component of the noun path.<br>**Event Type 1**<br>Event Type Name: `JDBC_Connection State`<br>*<noun type> State*<br>Event Type Path: `dms/JDBC/JDBC_Connection_State`<br>dms/*<leading component of noun path>*/*<noun type>*/*_State*<br>**Fields:**<br>■ ECID<br>■ RID<br>■ JDBC_Connection name<br>Value will be the full path of the noun<br>■ JDBC_Connection_Url<br>Value will be that of the state sensor of this name at the time of the event<br>■ JDBC_Connection_Username<br>Value will be that of the state sensor of this name at the time of the event<br>■ Event Name<br>Value will be one of the following:<br>■ The name of the DMS event sensor whose activation caused this JFR event instance<br>■ Null if this JFR event instance was created for a state sensor update |
| | Producer Name: JDBC<br>**Event Type 2**<br>Event Type Name: `JDBC_Connection CreateStatement`<br>Event Type Path:<br>`dms/JDBC/JDBC_Connection_CreateStatement`<br>**Fields:**<br>■ ECID<br>■ RID<br>■ JDBC_Connection name<br>■ JDBC_Connection_Url<br>■ JDBC_Connection_Username |

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

| DMS | JRockit Flight Recorder (JFR) |
|---|---|
| | Producer Name: JDBC<br>**Event Type 3**<br>Event Type Name: `JDBC_Connection CreateNewStatement`<br>Event Type Path:<br>`dms/JDBC/JDBC_Connection_CreateNewStatement`<br>**Fields:**<br>▪ ECID<br>▪ RID<br>▪ JDBC_Connection name<br>▪ JDBC_Connection_Url<br>▪ JDBC_Connection_Username |

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

| DMS | JRockit Flight Recorder (JFR) |
|---|---|
| | Producer Name: JDBC<br><br>**Event Type 4**<br>Event Type Name: `JDBC_Connection DBWaitTime`<br><br>Event Type Path:<br>`dms/JDBC/JDBC_Connection_DBWaitTime`<br><br>**Fields:**<br>■ ECID<br>■ RID<br>■ JDBC_Connection name<br>■ JDBC_Connection_Url<br>■ JDBC_Connection_Username |
| **Noun type:**<br>webcenter_lifecycle<br><br>**Noun path:**<br><br>`/oracle/webcenter/webcenter/lifecycle`<br><br>**Sensors:**<br>`ProcessingTime` (P)<br>`status` (S)<br>`successCount` (E)<br><br>**Where:**<br>P: Phase Sensor<br>S : State Sensor<br>E : Event Sensor | Producer Name: webcenter<br><br>**Event Type 1**<br>Event Type Name: `webcenter_lifecycle State`<br><br>**Fields:**<br>■ ECID<br>■ RID<br>■ webcenter_lifecycle name<br>■ status<br>■ event name |
| | Producer Name: webcenter<br>**Event Type 2**<br>Event Type Name: `webcenter_lifecycle ProcessingTime`<br>**Fields:**<br>■ ECID<br>■ RID<br>■ webcenter_lifecycle name<br>■ status |

## 5.7.3 Understanding DMS Event Output

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<br>For example:<br>**v1:**1280737384058:HTTP_REQUEST:STOP:/MyWebApp/emp |
| All | 2 | Event time | The time at which the event occurred<br>For example:<br>v1:**1280737384058**:HTTP_REQUEST:STOP:/MyWebApp/emp |
| All | 3 | Source object type | The type of object on which an action was performed to produce the event including:<br>■ NOUN<br>■ EVENT_SENSOR<br>■ STATE_SENSOR<br>■ PHASE_SENSOR<br>■ EXECUTION_CONTEXT<br>■ HTTP_REQUEST<br>For example:<br>v1:1280737384058:**HTTP_ 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:<br>■ CREATE<br>■ UPDATE<br>■ DELETE<br>■ START<br>■ STOP<br>■ ABORT<br>For example:<br>v1:1280737384058:HTTP_ REQUEST:**STOP**:/MyWebApp/emp |
| Nouns | 5 | Noun type | The name of the noun type<br>For example:<br>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<br>For example:<br>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<br>For example:<br>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<br>For example:<br>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<br>For example:<br>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.<br>For example:<br>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.<br>For example:<br>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:<br>■ State.DOUBLE<br>■ State.INTEGER<br>■ State.LONG<br>■ State.OBJECT<br>■ State.ANY<br>For example:<br>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.<br>For example:<br>v1:1280503318973:STATE_SENSOR:UPDATE:JDBC_ Connection:LogicalConnection:/JDBC/JDBC Data Source-0/CONNECTION_ 1:State.ANY:**LogicalConnection@13bed086** |
| HTTP Requests | 5 | URI | Uniform Resource Identifier (URI) identifies the resource upon which to apply the request.<br>For example:<br>v1:1280737382889:HTTP_ REQUEST:START:**/myWebApp/showEmployees**<br>v1:1280737384058:HTTP_ REQUEST:STOP:**/myWebApp/showEmployees** |
| Execution Context | 5 | ECID,RID | The context identifier (composed of ECID and RID separated by a comma).<br>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.<br>For example:<br>v1:1280737384058:EXECUTION_ CONTEXT:STOP:**bc4fd0668f79d507:367c127f:12a23f2013c:-8000-0000000000000f73,0** |

## 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 | - |
| Http 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 `java.lang.System.currentTimeMillis` to obtain times for PhaseEvents.<br>The default value for the units for the default clock is MSECS. |
| HIGHRES | The Java Highres clock uses `System.nanoTime()` (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.<br>**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.<br>**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.

# 6

# Oracle HTTP Server Performance Tuning

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



> **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 HTTP Server is based on the Apache 2.2.x infrastructure, and includes modules developed specifically by Oracle. The features of single sign-on, clustered deployment, and high availability enhance the operation of the Oracle HTTP Server.

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

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

## 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 HTTP Server, see "Monitoring Oracle HTTP Server Performance" in *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

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

In addition to the Fusion Middleware Control, Oracle HTTP Server also has Dynamic Monitoring Service (DMS), which collects metrics for every functional piece. You can review these metrics as needed to understand system behavior at a given point of time. This displays memory, CPU information and the 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.

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

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

   Examples:

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

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

## 6.3 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.

### 6.3.1 Tuning Oracle HTTP Server Directives

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

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

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

- Worker - This uses Multi-Process-Multi-Threads model and is the default MPM on all platforms other than Microsoft Windows platforms. Multi-thread support makes it more scalable by using fewer system resources and multi-process support makes it more stable.

- WinNT - This MPM is for Windows platforms only. It consists of a parent process and a child process. The parent process is the control process, and the child process creates threads to handle requests.
- Prefork - This is Apache 1.3.x style and uses processes instead of threads. This is considered the least efficient MPM.

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

---

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

---

***Table 6–1 Oracle HTTP Server Configuration Properties***

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

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

| Directive | Description |
|---|---|
| `ServerLimit`<br><br>Note that this parameter is not available in mod_winnt (Microsoft Windows). Winnt uses a single process, multi-threaded model | Specifies an upper limit on the number of server (child) processes that can exist or be created. This value overrides the `StartServers` value if that value is greater than the `ServerLimit` value. This is used to control the maximum number of server processes that can be created.<br><br>Default Value: 16 |
| `ThreadLimit` | Specifies the upper limit on the number of threads that can be created under a server (child) process. This value overrides the `ThreadsPerChild` value if that value is greater than the `ThreadLimit` value. This is used to control the maximum number of threads created per process to avoid conflicts/issues.<br><br>Default Values:<br><br>■ Windows Multi-Processing Module (mpm_winnt): 1920<br><br>■ All others: 64 |
| `ThreadsPerChild`<br><br>This directive maps to the **Threads Per Child Server Process** field on the Performance Directives screen. | Sets the number of threads created by each server (child) process at startup.<br><br>Default Value: 64 when mpm_winnt is used and 25 when Worker MPM is used.<br><br>The `ThreadsPerChild` directive works with other directives, as follows:<br><br>At startup, Oracle HTTP Server creates a parent process, which creates several child (server) processes as defined by the `StartServers` directive. Each server process creates several threads (server/worker), as specified in `ThreadsPerChild`, and a listener thread which listens for requests and transfers the control to the worker/server threads.<br><br>After startup, based on load conditions, the number of server processes and server threads (children of server processes) in the system are controlled by `MinSpareThreads` (minimum number of idle threads in the system) and `MaxSpareThreads` (maximum number of idle threads in the system). If the number of idle threads in the system is more than `MaxSpareThreads`, Oracle HTTP Server terminates the threads and processes if there are no child threads for a process. If the number of idle threads is fewer than `MinSpareThreads`, it creates new threads and processes if the `ThreadsPerChild` value has already been reached in the running processes.<br><br>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.<br><br>■ `ServerLimit` - Defines the upper limit on the number of servers that can be created. This affects `MaxClients` and `StartServers`.<br><br>■ `ThreadLimit` - Defines the upper limit on `ThreadsPerChild`. If `ThreadsPerChild` is greater than `ThreadLimit`, then it is automatically trimmed to the latter value.<br><br>■ `MaxClients` - Defines the upper limit on the number of server threads that can process requests simultaneously. This should be equal to the number of simultaneous connections that can be made. This value should be a multiple of `ThreadsPerChild`. If `MaxClients` is greater than `ServerLimit` multiplied by `ThreadsPerChild`, it is automatically be trimmed to the latter value. |

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

| Directive | Description |
|---|---|
| `MaxRequestsPerChild`<br>This directive maps to the **Max Requests Per Child Server Process** field on the Performance Directives screen. | Specifies the number of requests each child process is allowed to process before the child process dies. The child process ends to avoid problems after prolonged use when Apache (and any other libraries it uses) leak memory or other resources. On most systems, this is not needed, but some UNIX systems have notable leaks in the libraries. For these platforms, set `MaxRequestsPerChild` to 10000; a setting of 0 means unlimited requests.<br>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.<br>Default Value: 0<br>**Note**: On Windows systems `MaxRequestsPerChild` should always be set to `0` (unlimited) since there is only one server process. |
| `MaxSpareThreads`<br>`MinSpareThreads`<br>These directives map to the **Maximum Idle Threads** and **Minimum Idle Threads** fields on the Performance Directives screen.<br>Note that these parameters are not available in mod_winnt (Windows platform). | Controls the server-pool size. Rather than estimating how many server threads you need, Oracle HTTP Server dynamically adapts to the actual load. The server tries to maintain enough server threads to handle the current load, plus a few additional server threads to handle transient load increases such as multiple simultaneous requests from a single browser.<br>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.<br>Default Values:<br>MaxSpareThreads: 75<br>MinSpareThreads: 25 |
| `Timeout`<br>This directive maps to the **Request Timeout** field on the Performance Directives screen. | The number of seconds before incoming receives and outgoing sends time out.<br>Default Value: 300 |
| `KeepAlive`<br>This directive maps to the **Multiple Requests Per Connection** field on the Performance Directives screen. | Whether or not to allow persistent connections (more than one request per connection). Set to Off to deactivate.<br>Default Value: On |

*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.<br>If you have long client sessions, consider increasing this value.<br>Default Value: 100 |
| `KeepAliveTimeout`<br>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.<br>Default Value: 5 seconds |
| `limit`<br>`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.<br>Tuning the file descriptor limit can be accomplished by configuring the hard limit (`ulimit`) in a shell script which starts the OHS. Once the hard limit has been set the OHS will then adjust the soft limit (`limit`) to match.<br>Note that configuring file descriptor limits is platform specific. Refer to your operating system documentation for more information. |

## 6.3.2 Reducing Httpd 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 HTTP Server. Persistent Connections can improve performance by reducing the work load on the server. With Persistent Connections enabled, the server does not have to repeat the work to set up the connections with a client.

The default settings for the `KeepAlive` directives are:

```
KeepAlive on
MaxKeepAliveRequests 100
KeepAliveTimeOut 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 `ORACLE_HOME/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_osso` 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 HTTP Server performance.

### 6.4.1 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

#### 6.4.1.1 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 HTTP, and all major web browsers support SSL.

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

This section covers the following SSL performance-related information:

- Section 6.4.1.1.1, "Caching SSL on Oracle HTTP Server"

- Section 6.4.1.1.2, "Using SSL Application Level Data Encryption"
- Section 6.4.1.1.3, "Tuning SSL Performance"

**See Also:** *Oracle Fusion Middleware Application Security Guide*

**6.4.1.1.1 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 HTTP persistent connections. Oracle HTTP Server uses persistent HTTP connections by default, assuming they are supported on the client side. In HTTP over SSL as implemented by Oracle HTTP Server, SSL session state is kept in memory while the associated HTTP connection is persisted, a process which essentially eliminates the performance impacts associated with SSL session reuse (conceptually, the SSL connection is kept open along with the HTTP connection). For more information see Section 6.3.2, "Reducing Httpd Process Availability with Persistent Connections".

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

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

**6.4.1.1.3 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 HTTP to transfer the images. If the application requires server authentication only, do not use client authentication. If the performance goals of an application cannot be met by this method alone, additional hardware may be required.
2. Design the application to use SSL efficiently. Group secure operations to take advantage of SSL session reuse and SSL connection reuse.
3. Use persistent connections, if possible, to minimize cost of SSL session reuse.
4. Tune the session cache timeout value (the `SSLSessionCacheTimeout` directive in `ssl.conf`). A trade-off exists between the cost of maintaining an SSL session cache and the cost of establishing a new SSL session. As a rule, any secured business process, or conceptual grouping of SSL exchanges, should be completed without incurring session creation more than once. The default value for the `SSLSessionCacheTimeout` attribute is 300 seconds. It is a good idea to test an application's usability to help tune this setting.
5. If large volumes of data are being protected through SSL, pay close attention to the cipher suite being used. The `SSLCipherSuite` directive specified in `ssl.conf` controls the cipher suite. If lower levels of security are acceptable, use a less-secure protocol using a smaller key size (this may improve performance significantly). Finally, test the application using each available cipher suite for the specified security level to find the optimal suite.
6. If SSL remains a bottleneck to the performance and scalability of your application, after taking the preceding considerations into account, consider deploying multiple Oracle HTTP Server instances over a hardware cluster or consider the use of SSL accelerator cards.

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

---

## 6.4.2 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

### 6.4.2.1 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.

### 6.4.2.2 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
```

#### 6.4.2.3 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.

#### 6.4.2.4 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.

# 7

# Oracle Metadata Service (MDS) Performance Tuning

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



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

---

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

---

## 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 *Oracle Fusion Middleware Administrator's Guide*.

> **Note:** MDS Metrics, visible in Enterprise Manager, are useful for tuning the MDS cache. In particular, "`IOs Per MO Content Get`" or "`IOs Per Metadata Object Get`" should be less than 1. If not, consider increasing the size of the MDS cache. For more information on viewing DMS metric information, see Section 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. Unless time-to-live (TTL) is set, the MDS cache continues to occupy the `max-size-kb` of memory.

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

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

#### 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 *Oracle Fusion Middleware Administrator's Guide*.

### 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 *Oracle Fusion Middleware Administrator's Guide* ".

---

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

---

# 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 (HttpSession) until the user logs out.

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

# Part III

# Oracle Fusion Middleware Server Components

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

# 8

# Oracle Application Development Framework Performance Tuning

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

- Section 8.1, "About Oracle ADF"
- Section 8.2, "Basic Tuning Considerations"
- Section 8.3, "Advanced Tuning Considerations"

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

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

## 8.1 About Oracle ADF

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

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

## 8.2 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

### 8.2.1 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 8–1 describes some of these configuration options.

***Table 8–1 ADF Configuration Options***

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

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

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

### 8.2.2 Performance Considerations for ADF Faces

Table 8–2 provides configuration recommendations that may improve performance of ADF Faces:

***Table 8–2 Configuration Parameters for ADF Faces***

| Configuration Recommendation | Description |
|---|---|
| Avoid inline JavaScript in pages. | 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.<br>NOTE: Consider using af:resource rather than trh:script when possible. |
| Configure the JSP timeout parameter. | Using the JavaServer Pages (JSP) timeout parameter causes infrequently used pages to be flushed from the cache by the following setting in web.xml:<br>`<servlet>`<br>`<servlet-name>`<br>`oraclejsp`<br>`<init-param>`<br>`<param-name>`<br>`jsp_timeout`<br>`</param-name>`<br>`<param-value>`<br>`600`<br>`</param-value>`<br>`</init-param>`<br>`</servlet-name>`<br>`</servlet>`<br>NOTE: Set this parameter based on your own use case scenarios. |
| Create a single toolbar item with a drop-down popup. | 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.<br>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". |
| Remove unknown rowCount. | 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.<br>Remove unknown rowCount by setting DeferEstimatedRowCountProperty="false" on the view object (VO). |
| Disable pop-ups that cannot be displayed by the user. | 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.<br>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. |
| Do not use hover pop-ups on navigation links. | A hover popup on a navigation link causes the navigation to wait for the hover to be fetched first.<br>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. |

***Table 8–2 (Cont.) Configuration Parameters for ADF Faces***

| Configuration Recommendation | Description |
|---|---|
| Increase table scrolling timeout. | 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.<br><br>To prevent the performance impact, consider increasing the timeout limit to 300ms. |
| Use a timeout to call _ prepareForIncompleteImages. | 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.<br><br>For Internet Explorer 8 use a timeout (10ms) to call _ prepareForIncompleteImages so that the image tag called right after the .xml HTTP request is processed. Note that this is not an issue for Mozilla Firefox or Google Chrome. |
| Cache the GetFirstVisibleRowKeyandRow. | 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. |
| Use partial page navigation. | Partial Page Navigation is a feature of the ADF Faces framework that enables navigating from one ADF Faces page to another without a full page transition in the browser.The new page is sent to the client using Partial Page Rendering (PPR)/Ajax channel.<br><br>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.<br><br>Some known limitations of this feature are:<br><br>■ 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.<br><br>■ Applications cannot use anchor (hash) URLs for their own purposes. |

*Table 8–2 (Cont.) Configuration Parameters for ADF Faces*

| Configuration Recommendation | Description |
|---|---|
| Use page templates. | Page templates enable developers to build reusable, data-bound templates that can be used as a shell for any page. A developer can build one or more templates that provide structure and consistency for other developers building web pages. The templates have both static areas on them that cannot be changed when they are used and dynamic areas on them where the developer can place content specific to the page they are building.<br><br>There are some important considerations when using templates:<br><br>■ 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.<br><br>■ When building complex templates, sometimes it is easier to build them in multiple pieces and include them in the top-level template using `<f:subview>` tag. However, from a performance perspective, this is not typically recommended since it can impact memory usage on the server side. (`<f:subview>` introduces another level into the ID scoping hierarchy, which results in longer IDs. Long IDs have a negative impact on performance. Developers are advised to avoid using `<f:subview>` unless it is required. It is not necessary to use `<f:subview>` around `<jsp:include>` if you can ensure that all IDs are unique. For example, if you are using `<jsp:include>`, break a large page into multiple pieces for easier editing. And whenever possible, avoid using `<f:subview>`. If you are including content developed by someone else, use `<f:subview>` if you do not know which IDs the developer used. In addition, you do not have to put `<f:subview>` at the top of a region definition.<br><br>■ 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 8–2 (Cont.) Configuration Parameters for ADF Faces***

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

***Table 8–2 (Cont.) Configuration Parameters for ADF Faces***

| Configuration Recommendation | Description |
|---|---|
| Use ADF Rich Client Partial Page Rendering (PPR). | ADF Rich Client is based on Asynchronous JavaScript and XML (Ajax) development technique. Ajax is a web development technique for creating interactive web applications, where web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, without the whole web page being reloaded. The effect is to improve a web page's interactivity, speed, and usability.<br><br>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.<br><br>Two main Ajax patterns are implemented with partial page rendering (PPR):<br><br>■ native component refresh<br>■ cross-component refresh<br><br>While the framework builds in native component refresh, cross-component refresh has to be done by developers in certain cases.<br><br>Cross-component refresh is implemented declaratively or programmatically by the application developer defining which components are to trigger a partial update and which other components are to act as partial listeners, and so be updated. Using cross-component refresh and implementing it correctly is one of the best ways to improve client-side response time. While designing the UI page always think about what should happen when the use clicks a command button. Is it needed for the whole page to be refreshed or just an output text field? What should happen if the value in some field is updated? For more information, refer to *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*).<br><br>Consider a typical situation in which a page includes an `af:inputText` component, an `af:commandButton` component, and an `af:outputText` component. When the user enters a value for the `af:inputText`, then clicks the `af:commandButton`, the input value is reflected in the `af:outputText`. Without PPR, clicking the `af:commandButton` triggers a full-page refresh. Using PPR, you can limit the scale of the refresh to only those components you want to refresh, in this case the `af:outputText` component. To achieve this, you would do two things:<br><br>■ Set up the `af:commandButton` for partial submit by setting the `partialSubmit` attribute to `true`. Doing this causes the command component to start firing partial page requests each time it is clicked.<br>■ Define which components are to be refreshed when the partial submit takes place, in this example the `af:outputText` component, by setting the `partialTriggers` attribute for each of them to the id of the component triggering the refresh. In this example, this means setting the `partialTriggers` attribute of the `af:outputText` component to give the id of the `af:commandButton` component.<br><br>The steps above achieve PPR using a command button to trigger the partial page refresh.<br><br>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.<br><br>The ADF Rich Client has shown that partial page rendering results in the best client-side performance. Besides the impact on the client side, server-side processing can be faster and can have better server-side throughput and scalability. |

***Table 8–2 (Cont.) Configuration Parameters for ADF Faces***

| Configuration Recommendation | Description |
|---|---|
| Use ADF rich client navigation. | ADF Rich Client has an extensive support for navigation. One of the common use cases is tabbed navigation. This is currently supported by components like navigationPane which can bind to xmlMenuModel to easily define navigation.<br><br>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. |
| Cache resources. | Developers are strongly encouraged to ensure that any resources that can be cached (images, CSS, JavaScript) have their cache headers specified appropriately. Also, client requests for missing resources on the server result in addition round trips to the server. To avoid this, make sure all the resources are present on the server.<br><br>Consider using the ResourceServlet to configure web.xml to enable resource caching:<br><br>`<servlet-mapping>`<br>`  <servlet-name>resources</servlet-name>`<br>`  <url-pattern>/js/*</url-pattern>`<br>`</servlet-mapping>`<br>`<servlet-mapping>`<br>`  <servlet-name>resources</servlet-name>`<br>`  <url-pattern>/images/*</url-pattern>`<br>`</servlet-mapping>` |
| Reduce the size of state token cache | Property defined in web.xml org.apache.myfaces.trinidad.CLIENT_STATE_MAX_TOKENS in "token"-based client-side state saving, chooses how many tokens should be preserved at any one time. The default value is 15. When this is exceeded, state will have effectively been "forgotten" for the least recently viewed pages, which can impact users that actively use the Back button or that have multiple windows open simultaneously. 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 Back button this value should be set to 1. |

***Table 8–2 (Cont.) Configuration Parameters for ADF Faces***

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

***Table 8–2 (Cont.) Configuration Parameters for ADF Faces***

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

*Table 8–2 (Cont.) Configuration Parameters for ADF Faces*

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

## 8.2.3 Tuning ADF Faces Component Attributes

Table 8–3 provides configuration recommendations for ADF Faces Component Attributes:

***Table 8–3 ADF Faces Component Attributes***

| Configuration Recommendation | Description |
|---|---|
| Use the "immediate" attribute. | ADF Rich Client components have an `immediate` attribute. If a component has its `immediate` attribute set to `TRUE` (`immediate="true"`), then the validation, conversion, and events associated with the component are processed during the applyRequestValues phase. These are some cases where setting `immediate` to `TRUE` can lead to better performance.<br>■ The commandNavigationItem in the navigationPane can use the `immediate` attribute set to `TRUE` to avoid processing the data from the current screen while navigating to the new page.<br>■ If the input component value has to be validated before the other values, `immediate` should be set to `TRUE`. In case of an error it be detected earlier in the cycle and additional processing be avoided.<br>ADF Rich Client is built on top of JSF and uses standard JSF lifecycle. See "Understanding the JSF and ADF Faces Lifecycles" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*.<br>There are some important issues associated with the `immediate` attribute. Refer to "Using the Immediate Attribute" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for more information.<br>Note that this is an advanced feature. Most of the performance improvements can be achieved using the `af:subform` component. Refer to *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* for `af:subform` details. |
| Use the "visible" and "rendered" attributes. | All ADF Faces Rich Client display components have two properties that dictate how the component is displayed on the page:<br>■ The `visible` property specifies simply whether the component is to be displayed on the page, or is to be hidden.<br>■ The `rendered` property specifies whether the component shall exist in the client page at all.<br>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. |
| Use client-side events. | ADF Rich Client framework provides the client-side event model based on component-level events rather than DOM level. The client-side event model is a very useful feature that can speed up the application. Review the following performance considerations:<br>■ 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.<br>■ 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 8–3 (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. | ADF Rich Client framework has client-side components that play a role in client-side event handling and component behavior. The `clientComponent` attribute is used to configure when (or if) a client-side component should be generated. Setting `clientComponent` attribute to `TRUE` has a performance impact, so determine if its necessary to generate client-side components.<br>For more information, see "Client-side Components" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*. |
| Set the `childCreation` attribute on `af:popup` to `deferred` for a server-side performance enhancement | Setting `childCreation` to `deferred` 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.<br>CAUTION: This approach **should not be used** if any of the following tags are present inside the popup:<br>■ f:attribute<br>■ af:setPropertyListener<br>■ af:clientListener<br>■ af:serverListener<br>It also should not 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. |

### 8.2.4 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 8–4 provides some performance recommendations.

***Table 8–4 Table and Tree Component Configurations***

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

## 8.2.5 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.

## 8.2.6 Data Delivery - Lazy versus Immediate

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

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

When choosing between these two options, consider the following:

| | |
|---|---|
| Lazy Delivery (default) | Lazy delivery should be used 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.<br><br>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.<br><br>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 `RenderHint` on the iterator to *background*.<br><br>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.<br><br>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. |
| Immediate Delivery | Immediate delivery (`contentDelivery="immediate"`) should be used if table data control is fast, or if it returns a small set of data. In these cases the response time be faster than using lazy delivery.<br><br>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. |

### 8.2.7 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 8–5 provides some configuration recommendations for DVT components:

*Table 8–5 DVT Component Configurations*

| Configuration Recommendation | Description |
|---|---|
| Modify the RangeSize attribute. | The RangeSize attribute defines the number of rows to return simultaneously. A RangeSize value of -1 causes the iterator to return all the rows. Using a lower value may improve performance, but it may be harder to stop the data and any data beyond rangeSize is not available in the view. |
| Use horizontal text instead of vertical text. | By default, pivot tables use horizontal text for column headers. However, there is an option to use vertical text as well. Vertical text can be used by specifying a CSS style for the header format such as:<br>`writing-mode:tb-rl;filter:flipV flipH;`<br>While vertical text can look better in some cases, it has a performance impact when the Firefox browser is used.<br>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.<br>To have the best performance, consider using horizontal text instead of vertical text. |

# 8.3 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.

## 8.3.1 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.

### 8.3.1.1 HTTP 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 HTTP 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>
```

#### 8.3.1.2 View Objects Tuning

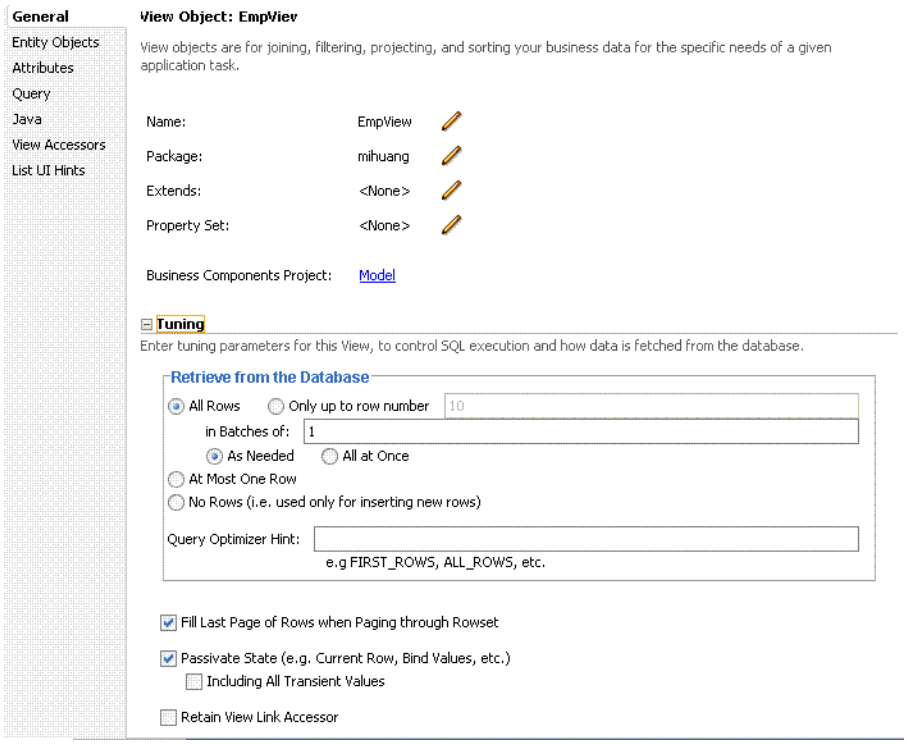
View objects (VOs) provide many tuning options to enable a developer to tailor the View Object to the application's specific needs. View Objects should be configured to use the minimal feature set required to fulfill the functional requirement. The *Oracle Fusion Middleware Fusion Developer's Guide for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

**8.3.1.2.1 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 | 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:<br>■ Non-updatable EO-based View Objects<br>■ Expert-mode View Objects<br>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.<br>Expert-mode View Objects have the ability to perform SQL operations not supported by EOs and avoid the small performance impact from coordinating View Object and EO rows. EO-based View Objects can be marked non-updatable by deselecting the "updatable" option in the selected EO for the View Object, which can also be done by adding the parameter `ReadOnly="true"` on the `EntityUsage` attribute in the View Object XML definition. |
| Insert-only View Objects | For View Objects that are used only for inserting records, you can prevent unnecessary select queries from being executed when using the View Object. To do this, set the option No Rows in the Retrieve from the Database group box in the View Objects Overview tab. This sets MaxFetchSize to 0 (zero) for the View Object definition. |
| run time-created View Objects | View Objects can be created at run time using the `createViewObjectFromQueryStmt()` API on the AM. However, avoid using run time-created View Objects unless absolutely necessary due to potential performance impacts and complexity of tuning. |

**8.3.1.2.2 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 8–1.

***Figure 8–1 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.<br><br>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.<br><br>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.<br><br>Max fetch size can also be used to limit the impact from an non-selective query that may return hundreds (or thousands) of rows. In such cases, specifying the max fetch size limits the number of rows that can be fetched and stored into memory. |

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

#### 8.3.1.2.3 Additional View Object Configurations

Table 8–6 provides additional tuning considerations when using the View Object:

***Table 8–6 Additional View Object Configurations***

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

### 8.3.1.3 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"`).

#### 8.3.1.4 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.

#### 8.3.1.5 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
  ```

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

#### 8.3.1.6 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. For detailed explanations of the different AM pool configurations, see " Tuning Application Module Pools" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

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

***Table 8–7 Application Module (AM) Pool Tuning***

| Configuration Recommendation | Description |
|---|---|
| Optimize the number of AM pools in the application. | Parameters applied at the system level are applied per AM pool. If the application uses more than 1 AM pool, then system-level values for the number of AM instances must be multiplied by the number of AM pools to realize the actual limits specified on the system as a whole. For instance, if an application uses 4 separate AM pools to service the application and a system-level configuration is used to limit the max AM pool size to 100, then this can result in a maximum of 400 AM instances (4 pools * 100 max pool size). If the intent is to limit the entire application to a max pool size of 100, then the system-level configuration should specify a max pool size of 25 (100 max pool size / 4 pools). Finer granularity for configuring each AM pool can be achieved by configuring each pool separately through JDev or directly in bc4j.xcfg. |
| Optimize the number of database connections. | By default AM instances retain their database connections even when checked back into the AM pool. There are many performance benefits to maintain this association. To maintain performance, consider configuring more AM instances than the maximum number of specified database connections.<br><br>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.<br><br>For more information see "Setting Pool Configuration Parameters" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*. |

**8.3.1.6.1 General AM Pool Configurations** The following guidelines can be used as a general starting point when tuning AM and AM pool behavior. Details for each parameter can be found in the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*. More specific tuning for memory or CPU usage can be found in Section 8.3.1.6.2, "AM Pool Sizing Configurations".

***Table 8–8 AM Pool Tuning Parameters***

| Parameter | Description |
|---|---|
| `jbo.ampool.initpoolsize` | Specifies the number of application module instances to create when the pool is initialized (default is zero). Setting a nonzero initial pool size increases the time to initialize the application, but improves subsequent performance for operations requiring an AM instance. A general guideline is to configure this to 10% more than the anticipated number of concurrent AM instances required to service all users. |
| `jbo.ampool.maxpoolsize` | Specifies the maximum number of application module instances that the pool can allocate (default is 4096). The pool can never create more application module instances than this limit imposes. A general guideline is to configure this to 20% more than the initial pool size to allow for some additional growth. |
| `jbo.ampool.minavailablesize` | Specifies the minimum number of available application module instances that the pool monitor should leave in the pool during a resource cleanup operation (default is 5). The ideal minimum value for this configuration should be at least 1 to avoid the costs of re-creating the AM pool. Setting this to zero (0) can cause the pool itself to be cleaned up when all instances have been idle for longer than the idle time out. |
| `jbo.ampool.maxavailablesize` | Specifies the ideal maximum number of application module instances in the pool when not under abnormal load (default is 25). When the pool monitor wakes up to do resource cleanup, it tries to remove available application module instances to bring the total number of available instances down to this ideal maximum. Instances that have not been used for a period longer than the idle instance time out is cleaned up at this time, and then additional available instances can be removed if necessary to bring the number of available instances down to this size. |
| `jbo.recyclethreshold` | Specifies the maximum number of application module instances in the pool that attempt to preserve session affinity for the next request made by the session that used them last before releasing them to the pool in managed-state mode (default is 10). The referenced pool size should always be less than or equal to the maximum pool size. This enables the configured number of available instances to try and remain "loyal" to the affinity they have with the most recent session that released them in managed state mode. A general guideline is to configure this to the expected number of concurrent users that perform multiple operations with short think times. If there are no users expected to use the application with short think times, then this can be configured to 0 (zero) to eliminate affinity. |
| `jbo.ampool.timetolive` | Specifies the number of milliseconds that an application module instance lives in the pool. After this time, the instance is a candidate for removal during the next resource cleanup regardless of whether it would bring the number of instances in the pool below `minavailablesize` The default is 3600000ms or 1 hour. The default value is sufficient for most applications. |
| `jbo.ampool.maxinactiveage` | Specifies the number of milliseconds after which to consider an inactive application module instance in the pool as a candidate for removal during the next resource cleanup (default is 600000ms = 10 minutes). |
| `jbo.ampool.monitorsleepinterval` | Specifies the length of time in milliseconds between pool resource cleanup (default is 600000ms = 10 minutes). While the number of application module instances in the pool should never exceed the maximum pool size, available instances that are candidates for removal from the pool do not get "cleaned up" until the next time the application module pool monitor wakes up to do its job. |

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

| Parameter | Description |
|---|---|
| jbo.dofailover | Specifies whether to disable or enable failover. By default, failover is disabled. To enable failover, set the parameter to true.<br><br>**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 SQLException (Connection has already been closed) that is saved to the server log. The exception is not reported through the user interface. To ensure that state passivation occurs and changes are saved, set an appropriate value for the weblogic-application.xml deployment descriptor parameter inactive-connection-timeout-seconds on the <connection-check-params> pool-params element. 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. |
| jbo.locking.mode | Specifies the locking mode (optimistic or pessimistic). The default is pessimistic, which means that a pending transaction state can be created on the database with row-level locks. With pessimistic locking mode, each time an AM is recycled, a rollback is issued in the JDBC connection. Web applications should set the locking mode to optimistic to avoid creating the row-level locks. |
| jbo.doconnectionpooling | Specifies whether the AM instance can be disconnected from the database connection when the AM instance is returned to the AM pool. This enables an application to size the AM pool larger than the database connection pool. The default is false, which means that an AM instance can retain its database connection when the AM instance is returned to the AM pool. When set to true, the AM can release the database connection back to the database connection pool when the AM instance is returned to the AM pool. Note that before an AM is disconnected from the database connection, a rollback can be issued on that database connection to revert any pending database state. |
| jbo.txn.disconnect_level | When used in conjunction with jbo.doconnectionpooling=true, specifies BC4J behavior for maintaining JDBC ResultSets. By default jbo.txn.disconnect_level is 0, and passivation can be used to close any open ResultSets when the database connection is disconnected from the AM instance. Configuring jbo.txn.disconnect_level to 1 can prevent this behavior to avoid the passivation costs for this situation. |

#### 8.3.1.6.2 AM Pool Sizing Configurations

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

For parameters that can be configured for memory-constrained systems, see Table 8–9.

*Table 8–9 AM Pool Sizing Configurations - Memory Considerations*

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

For parameters that can be configured to reduce the load on the CPU to some extent through a few parameters, see Table 8–10.

***Table 8–10 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. |

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

For memory-constrained systems, configure the AM pool to clean up more AM instances more frequently so that the memory consumed by the AM instance can be freed for other purposes. However, reducing the number of available AM instances and increasing the frequency of cleanups can result in higher CPU usage and longer response times. See Table 8–11 for more information.

***Table 8–11 AM Pool Resource Cleanup Configurations - Memory Considerations***

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

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

*Table 8–12 AM Pool Resource Cleanup Configurations - CPU Considerations*

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

#### 8.3.1.7 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.

#### 8.3.1.8 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.

#### 8.3.1.9 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

#### 8.3.1.10 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.

#### 8.3.1.11 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 *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

#### 8.3.1.12 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 *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

# 9

# Oracle TopLink (EclipseLink) JPA Performance Tuning

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

- Section 9.1, "About Oracle TopLink and EclipseLink"
- Section 9.2, "Basic Tuning Considerations"
- Section 9.3, "Advanced Tuning Considerations"

---

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

- EclipseLink Performance Tuning at `http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Performance`
- Performance Monitoring and Profiling at `http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Performance/Performance_Profiling`
- Introduction to Optimization at `http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Performance#Identifying_General_Performance_Optimization`

---

## 9.1 About Oracle TopLink and EclipseLink

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

The Java Persistence API is a specification for persistence in Java EE and Java SE applications. In JPA, a persistent class is referred to as an entity. An entity is a plain old Java object (POJO) class that is mapped to the database and configured for usage through JPA using annotations, persistence XML, or both. This chapter focuses on tuning JPA in the context of EJB3.0 and a Java EE environment.

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

- The EclipseLink JPA User's Guide at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA
- "Considering JPA Entity Architecture" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Introduction/Architecture
- Introduction to EclipseLink Queries at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Querying
- Introduction to Cache at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Caching
- Introduction to Mapping and Configuration at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Mapping

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

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

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

## 9.2 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

### 9.2.1 Creating Efficient SQL Statements and Queries

This section covers using efficient SQL statements and SQL querying. Table 9–1 and Table 9–2 show tuning parameters and performance recommendations related to SQL statements and querying.

***Table 9–1 EJB/JPA Using Efficient SQL Statements and Querying***

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

*Table 9–1 (Cont.) EJB/JPA Using Efficient SQL Statements and Querying*

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| Fetch Size | The JDBC fetch size gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed.<br>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.<br>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.<br>Note: The default value means use the JDBC driver default value, which is typically 10 rows for the Oracle JDBC driver.<br>To configure this, use query hint `"eclipselink.jdbc.fetch-size"`.<br>Default Value: 0 | The optimal fetch size is not always obvious. Usually, a fetch size of one half or one quarter of the total expected result size is optimal. Note that if you are unsure of the result set size, incorrectly setting a fetch size too large or too small can decrease performance. |
| Batch Writing | Batch writing can improve database performance by sending groups of `INSERT`, `UPDATE`, and `DELETE` statements to the database in a single transaction, rather than individually.<br>The persistence property in persistence.xml `"eclipselink.jdbc.batch-writing"="JDBC"` is used to configure this.<br>Default Value: Off | Enable for the persistence unit. |
| Change Tracking | This is an optimization feature that lets you tune the way EclipseLink detects changes in an Entity.<br>Default Value: AttributeLevel if using weaving (Java EE default), otherwise Deferred. | Leave at default AttributeLevel for best performance. |
| Weaving | Can disable through persistence.xml properties `"eclipselink.weaving"`<br>Default Value: On | Leave on for best performance. |

*Table 9–1 (Cont.) EJB/JPA Using Efficient SQL Statements and Querying*

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

#### 9.2.1.1 Tuning Entity Relationships Query Parameters

Table 9–2 shows the Entity relationship query parameters for performance tuning.

***Table 9–2 EJB3.0 Entity Relationship Query Performance Options***

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| Batch Fetching | 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.<br><br>Batch fetching has three types: JOIN, EXISTS and IN. The type is set through the query hint "eclipselink.batch.type"<br><br>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.<br><br>Default Value: Off | 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.<br><br>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. |
| Join Fetching | 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.<br><br>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.<br><br>You can specify the use of join in `JPQL (JOIN FETCH)`, or you can set it multi-level in a query hint, "eclipselink.join-fetch". It also can be set in the mapping annotation @JoinFetch.<br><br>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.<br><br>See Also: "Join Fetch" at `http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Querying/Query_Hints#Join_Fetch`<br><br>Default Value: Not Used | 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 |

*Table 9–2 (Cont.) EJB3.0 Entity Relationship Query Performance Options*

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| Lazy loading | Without lazy loading on, when EclipseLink retrieves a persistent object, it retrieves all of the dependent objects to which it refers. When you configure lazy reading (also known as indirection, lazy loading, or just-in-time reading) for an attribute mapped with a relationship mapping, EclipseLink uses an indirection object as a place holder for the referenced object.<br><br>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.<br><br>See Also: "Lazy Loading" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Mapping/Basic_Mappings/Lazy_Basics<br><br>Default Value: On for collection mapping (ToMany mappings, @OneToMany, @ManyToMany)<br><br>Default Value: Off for reference (ToOne mappings, @OneToOne, @ManyToOne)<br><br>(Note that setting lazy loading On for @OneToOne, @ManyToOne requires weaving, which is On by default for Java Java EE.) | 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.<br><br>You may also consider using optimized loading with LoadGroups which allows a query to force instantiation of relationships. |

## 9.2.2 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 9.3.1, "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 9–3 shows the cache configuration options.

For more information on cache configuration, see "Caching" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Caching.

**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 9–3.

*Table 9–3 EJB3.0 JPA Entities and Cache Configuration Options*

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| Object Cache | EclipseLink sessions provide an object cache. EJB3.0 JPA applications that use the EclipseLink persistence manager create EclipseLink sessions that by default use this cache. This cache, known as the **session cache**, retains information about objects that are read from or written to the database, and is a key element for improving the performance of an EclipseLink application.<br>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.<br>The annotation type @Cacheable specifies whether an entity should be cached. Caching is enabled when the value of the persistence.xml caching element is ENABLE_SELECTIVE or DISABLE_SELECTIVE. The value of the Cacheable annotation is inherited by subclasses; it can be overridden by specifying Cacheable on a subclass.<br>Cacheable(false) means that the entity and its state must not be cached by the provider.<br>Default Value: Enabled (shared is True) | Generally it is recommended that you leave caching enabled. If you have an object that is always read from the database, as in a pessimistic locked object, then the cache for that entity should be disabled. Also, consider disabling the cache for infrequently accessed entities |
| Query Result Set Cache | In addition to the object cache in EclipseLink, EclipseLink also supports a query cache:<br>▪ 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.<br>▪ 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.<br>The query hints for a query cache are:<br>`"eclipselink.query-cache"`<br>`"eclipselink.query-cache.size"`<br>`"eclipselink.query-cache.invalidation"`<br>See Also: "Caching" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Caching and "EclipseLink JPA Query Hints" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Querying/Query_Hints<br>Default Value: Not Used | Use for frequently executed non-primary key queries with infrequently changing result sets.Use with a cache invalidation time out to refresh as needed. |

*Table 9–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options*

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| Cache Size | Cache size can be configured through persistence properties:<br>`"eclipselink.cache.size.<entity>"`<br>`"eclipselink.cache.size.default"`<br>`"eclipselink.cache.type.default"`<br>See Also: "Configuring Persistence Units Using persistence.xml" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Configuration/JPA/persistence.xml and 'Class PersistenceUnitProperties" at http://www.eclipse.org/eclipselink/api/2.3/org/eclipse/persistence/config/PersistenceUnitProperties.html<br>Default Value: Type `SoftWeak`, Size 100 (per Entity). | Set the cache size relative to how much memory you have available, how many instances of the class you have, the frequency the entities are accessed, and how much caching you want based on your tolerance for stale data.<br>Consider creating larger cache sizes for entities that have many instances that are frequently accessed and stale data is not a big issue.<br>Consider using smaller cache sizes or no cache for frequently updated entities that must always have fresh data, or infrequently accessed entities. |
| Locking | Oracle supports the locking policies shown in Table 9–4: no locking, optimistic, pessimistic, and read-only.<br>Locking is set through JPA @Version annotation, eclipselink.read-only<br>How to Use EclipseLink Locking at http://wiki.eclipse.org/EclipseLink/Examples/JPA/Locking<br>Default Value: No Locking | For entities that can be updated concurrently, consider using the locking policy to prevent a user from writing over another users changes. To optimize performance for read-only entities, consider defining the entity as read-only or use a read-only query hint. |

***Table 9–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options***

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

*Table 9–3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options*

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| Cache Refreshing | By default, EclipseLink caches objects read from a data source. Subsequent queries for these objects access the cache and thus improve performance by reducing data source access and avoiding the cost of rebuilding object's and their relationships. Even if a query accesses the data source, if the objects corresponding to the records returned are in the cache, EclipseLink uses the cached objects. This default caching policy can lead to stale data in the application.<br>Refreshing can be enabled at the entity level (`alwaysRefresh` or `refreshOnlyIfNewer` and `expiry`) and at the query level (with the `eclipselink.refresh` query hint). You can also force queries to go to the database with (`disableHits`). Using an appropriate locking policy is the only way to ensure that stale or conflicting data does not get committed to the database.<br>For more information see: Section 9.2.2.1, "Cache Refreshing Scenarios"<br>See Also: "Caching Overview" at `http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Caching/Caching_Overview`<br>Default Value: No Cache Refreshing | Try to avoid entity level cache refresh and instead, consider configuring the following:<br>■ cache refresh on a query-by-query basis<br>■ cache expiration<br>■ isolated caching |

### 9.2.2.1 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.

### 9.2.2.2 Tuning the Locking Mode Policies

The locking modes, as shown in Table 9–4, 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 "Locking" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Mapping/Locking.

***Table 9–4 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.<br>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.<br>See Also: "Optimistic Locking" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Mapping/Locking/Optimistic_Locking | If infrequent concurrent updates to the same rows are expected, then optimistic locking may provide the best performance while providing data consistency guarantees. |
| Pessimistic | The first user who accesses the data with the purpose of updating it locks the data until completing the update. | 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.<br>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.<br>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 |

## 9.2.3 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, "Configuring Mappings" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Basic_JPA_Development/Mapping.

### 9.2.4 Using Data Partitioning

EclipseLink allows you to configure data partioning using the @Partitioned annotation. Partitioning enables an application to scale information across multiple databases; including clustered databases. For more information on using @Partioned and other partitioning policy annotations, see "Data Partitioning" at http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Data_Partitioning.

## 9.3 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

### 9.3.1 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 "JPA on the Grid" Approach at http://www.oracle.com/technology/products/ias/toplink/doc/11110/grid/tlgug003.htm

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

### 9.3.2 Analyzing EclipseLink JPA Entity Performance

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

- Form monitoring performance, see "Performance Monitoring" in the EclipseLink User's Guide. Note that this tool is intended to profile and monitor information in a multithreaded server environment.
- For profiling performance, see "Measuring EclipseLink Performance with the EclipseLink Profiler" in the EclipseLink User's Guide. Note that this tool is intended for use with single-threaded finite use cases.

- For debugging performance issues and testing, you can view the SQL generated from EclipseLink. To view the SQL, increase the logging level to "FINE" by using the EclipseLink JPA extensions for logging.

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

# 10 Oracle Web Cache Performance Tuning

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



## 10.1 About Oracle Web Cache

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

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

## 10.2 Performance Considerations

Effective Oracle Web Cache performance tuning starts with a good understanding of its usage and general performance issues. Before you begin tuning Oracle Adaptive Access Manager, review this section as well as the recommendations discussed in Top Performance Areas:

- Optimizing Hardware Resources
- Optimizing Platform Connections

### 10.2.1 Optimizing Hardware Resources

- Hardware Resources
- Memory Configuration

#### 10.2.1.1 Hardware Resources

Oracle Web Cache performs best with one very powerful CPU or two CPUs. Because Oracle Web Cache is an in-memory cache, it is rarely limited by CPU cycles. Additional CPUs do not increase performance significantly. However, the speed of the

processors is critical-use the fastest CPUs you can afford. Use more CPUs if Web Cache is sharing the system with other Oracle application server components or other applications.

Note that Oracle Web Cache is limited by the available addressable memory. Additional memory can increase performance and scalability. For information about the amount of memory needed, see Section 10.2.1.2, "Memory Configuration".

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

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

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

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

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

#### 10.2.1.2 Memory Configuration

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

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

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

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

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

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

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

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

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

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

### 10.2.2 Optimizing Platform Connections

- UNIX Connections
- Windows Connections

#### 10.2.2.1 UNIX Connections

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

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

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

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

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

#### 10.2.2.2 Windows Connections

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

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

## 10.3 Basic Tuning Considerations

This section provides the basic tuning considerations for Oracle Web Cache. It contains the following tuning recommendations:

- Optimizing Network Connections
- Increasing Cache Hit Rates
- Optimizing Response Time

### 10.3.1 Optimizing Network Connections

- Network Bandwidth
- Network Connections
- Network-Related Parameters

#### 10.3.1.1 Network Bandwidth

When you use Oracle Web Cache, ensure that each system has sufficient network bandwidth to accommodate the throughput load. Otherwise, the network may be

saturated but Oracle Web Cache has additional capacity. For example, if an application generates 100 megabits of data or more per second, 10/100 Megabit Ethernet can be saturated.

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

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

#### 10.3.1.2 Network Connections

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

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

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

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

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

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

#### 10.3.1.3 Network-Related Parameters

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

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

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

- Keep-Alive Timeout

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

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

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

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

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

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

- Origin Server Timeout

  The amount of time for the application Web server to generate a response to Oracle Web Cache. If the application Web server or proxy server is unable to generate a response within that time, Oracle Web Cache sends a network apology page to the browser.

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

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

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

- `KeepAlive`

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

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

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

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

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

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

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

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

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

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

This setting is usually only an issue during stress testing, if you continuously open more TCP/IP connections from one client computer. In this situation, lower the TCP time-wait setting. In real world deployments, this is rarely an issue because it is unlikely that a single client can generate a huge number of connections.

### 10.3.2 Increasing Cache Hit Rates

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

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

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

  Oracle Web Cache can cache different versions of a document with the same URL, based on request cookies or headers. To use this feature, applications may need to

implement simple changes, such as creating a cookie or header that differentiates the documents.

Some applications contain insignificant URL parameters, which can lead to different URLs representing the same content. If the documents are cached under their full URLs, the cache hit/miss ratio becomes very low. You can configure Oracle Web Cache to ignore the non-differentiating URL parameter values, so that a single document is cached for different URLs, greatly increasing cache hit rates.

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

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

- Use redirection to cache entry documents.

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

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

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

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

- Use partial page caching where possible.

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

  Depending on the application, updating a fragment can be cheaper than updating a full page. In addition, each template or fragment can have its own unique caching policies such as expiration, validation, and invalidation, so that each

fragment in a full Web page can be cached if possible, even when some fragments are not cached or are cached for a much shorter period of time.

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

  Personalized information often appears in Web pages, making them unique for each user. For example, many Web pages contain tens or hundreds of hyperlinks embedding application session IDs. To resolve this, create your ESI pages with variables. Because variables can resolve to different pieces of request information or response information, the uniqueness of templates and fragments can be significantly reduced. This, in turn, results in better cache hit/miss ratios.

### 10.3.3 Optimizing Response Time

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

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

First, check the following:

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

Then, check the following:

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

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

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

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

- The number of cache connections. Check Maximum Incoming Connections in the Resource Limits page (Properties > Resource Limits).
- The memory size for the cache. Check Maximum Cache Size in the Resource Limits page (Properties > Resource Limits).

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

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

- Network bandwidth
- CPU usage

## 10.4 Advanced Tuning Considerations

The following Oracle Web Cache tuning considerations are provided as a guide. Always consult your own use case scenarios to determine if these configurations should be used in your deployment.

### 10.4.1 Optimizing Performance with Oracle ADF

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

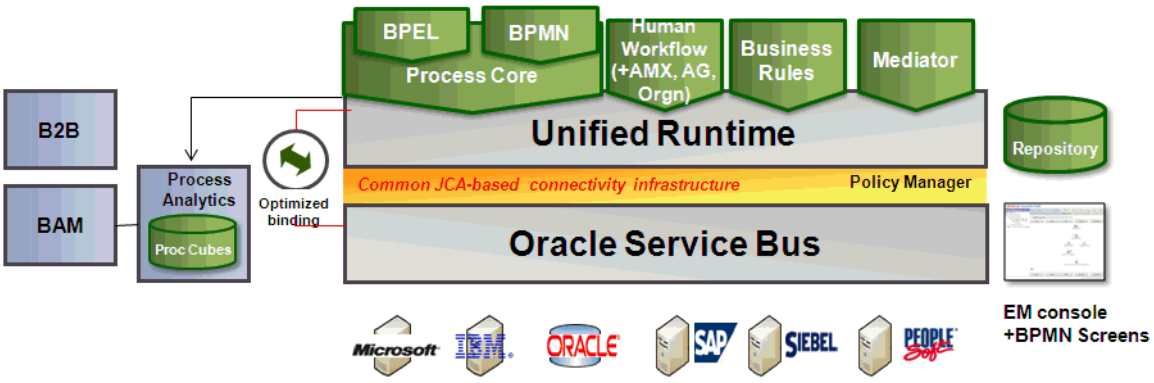
- After you configure the Maximum Cache Size setting in the Resource Limits page of Oracle Web Cache Manager, use a simulated load or an actual load to monitor the cache to see how much memory is actually used. Verify that any additional memory usage does not result in the host swapping memory to disk,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

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

# Part IV

## SOA Suite Components

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




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

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

The SOA Suite Components are documented in the following chapters:

- Chapter 11, "General Tuning for SOA Suite Components"
- Chapter 12, "Oracle Business Rules Performance Tuning"
- Chapter 13, "Oracle BPEL Process Manager Performance Tuning"
- Chapter 15, "Oracle Mediator Performance Tuning"
- Chapter 16, "Oracle Business Process Management Performance Tuning"
- Chapter 17, "Oracle Human Workflow Performance Tuning"
- Chapter 18, "Oracle Adapters Performance Tuning"
- Chapter 14, "Oracle Business Activity Monitoring Performance Tuning"
- Chapter 19, "User Messaging Service Performance Tuning"
- Chapter 20, "Oracle B2B Performance Tuning"

# 11

# General Tuning for SOA Suite Components

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

- Section 11.1, "About SOA Suite Configuration Properties"
- Section 11.2, "SOA Infrastructure Configurations"
- Section 11.3, "Modifying SOA Configuration Parameters"
- Section 11.4, "Tuning JVM for SOA Performance"
- Section 11.5, "Tuning Database Settings for SOA Performance"

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

> **Note:** Additional SOA tuning recommendations can be found in "Managing Large Documents and Large Number of Instances" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

## 11.1 About SOA Suite Configuration Properties

Refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* for more information on configuring the SOA Applications.

## 11.2 SOA Infrastructure Configurations

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

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

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

### 11.2.1 Audit Level

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

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

### 11.2.2 Composite Instance State

You can use the `CompositeInstanceStateEnabled` property to configure the SOA composite application instance state. Note, however, that enabling this option may impact performance during instance processing. This option enables separate tracking of the running instances. All instances are captured as either running or not running. This information displays later in the State column of the composite instances tables for the SOA Infrastructure and SOA composite application. The valid states are running, completed, faulted, recovery needed, stale, terminated, suspended, and state not available.

### 11.2.3 instanceTrackingAuditTrailThreshold

This parameter is used to limit the audit trail size while it is being built. The default value is 1MB. If the audit trail exceeds the `instanceTrackingAuditTrailThreshold` size (1MB by default), then an exception is thrown, and the audit trail is not fully built. The value is in Bytes, so the default value is 1024*1024. This parameter can improve performance, as it prevents huge audit trails to potentially consume a lot or all the memory available on the SOA server where the audit trail is built. So in many way it acts as a safety valve. However in some cases users might want to increase the default value, if they get an exception while retrieving audit trails from Enterprise Manager, that states the " instanceTrackingAuditTrailThreshold" has been exceeded.

### 11.2.4 Logging Level

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

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

## 11.3 Modifying SOA Configuration Parameters

SOA soa-infra level configurations can be set through Oracle Enterprise Manager.

For more information, see "Getting Started with Administering Oracle SOA Suite and Oracle BPM Suite" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

## 11.4 Tuning JVM for SOA Performance

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

## 11.5 Tuning Database Settings for SOA Performance

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

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

### 11.5.1 Configuring Data Sources for SOA

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

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

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

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

### 11.5.2 Managing Tables and Indexes

Consider using hash partitioning on your tables and indexes if your data does not easily lend itself to range partitioning, but you would like to partition for performance and manageability reasons. Hash partitioning provides a method of evenly distributing data across a specified number of partitions. Rows are mapped into partitions based on a hash value of the partitioning key. Creating and using hash partitions gives you a highly tunable method of data placement, because you can influence availability and performance by spreading these evenly sized partitions across I/O devices (striping).

To improve performance, consider using hash partitioning on the following tables and indexes:

| Partitioned Table Name | Partition Type | Number |
|---|---|---|
| AUDIT_COUNTER | Hash partitioning of AC_PK index | Not Applicable |
| CUBE_INSTANCE | Partitioned and Reverse key index CI_CREATION_DATE | Not Applicable |
| CUBE_SCOPE | Partition by hash (CIKEY) | Partitions = 200 |
| MEDIATOR_CASE_INSTANCE | Partition by hash ("ID") | Partitions = 200 |
| XML_DOCUMENT | Partition by hash (document_id) | Partitions = 200 |

| Hash Partitioned Indexes |
|---|
| COMPOSITE_INSTANCE_CREATED |
| BRDECISIONINSTANCE_INDX3 |
| MEDIATOR_INSTANCE_INDEX2 |
| MEDIATOR_INSTANCE_INDEX5 |
| MEDIATOR_INSTANCE_INDEX6 |
| MEDIATOR_INSTANCE_INDEX1 |
| MEDIATOR_INSTANCE_INDEX3 |
| MEDIATOR_CASE_INSTANCE_INDEX2 |
| MEDIATOR_CASE_DETAIL_INDEX1 |
| REFERENCE_INSTANCE_CO_ID |
| CI_NAME_REV_STATE |
| DOC_DLV_MSG_GUID_INDEX |
| STATE_TYPE_DATE |

### 11.5.3 Tuning Weblogic Server Performance for SOA

For optimum performance, you must ensure that the WebLogic  For complete performance tuning of Weblogic Server, refer to *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

# 12

# Oracle Business Rules Performance Tuning

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

The chapter includes the following sections:



## 12.1 About Oracle Business Rules

Oracle Business Rules provides high performance and easy to use implementation of Business Rules technology. It provides 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.

## 12.2 Basic Tuning Considerations

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.



### 12.2.1 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).

### 12.2.2 Assert Child Facts instead of Multiple Dereferences

Expressions like `Account.Contact.Address` 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 `assertXPath` or other means to assert object hierarchy; that is for the preceding example, assert both Account and Contact as Fact Types.

### 12.2.3 Avoid Side Affects in Rule Conditions

Methods or functions that have side affects such as changing a value or state should not be used in a rule condition. Due to the optimizations performed when the rule engine builds the Rete network, and the Rete network operations that are performed as facts are asserted, modified (and re-asserted), or retracted, the tests in a rule condition may be evaluated a greater or lesser number of times than would occur in a procedural program. If a method or function has side effects, those side effects may be performed an unexpected number of times.

### 12.2.4 Avoid Expensive Operations in Rule Conditions

Expensive operations should be avoided in rule conditions. Expensive operations would include any operation that involves I/O (disk or network) or even intensive computations. In general, consider avoiding I/O or DBMS access from the rules engine directly. These operations should be done external to the rules engine. 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.

### 12.2.5 Consider Pattern Ordering

Reordering rule patterns can improve the performance of rule evaluation in time, memory use, or both. There are two main guidelines for ordering fact clauses (patterns) within a rule condition.

- If a fact is not expected to change (or will not change frequently) during rule evaluation, place its fact clause before fact clauses that change more frequently. That is, order the fact clauses by expected rate of change from least to greatest. Ordering fact clauses in this way can improve the performance (time) of rule evaluation.
- 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, place that fact clause before the others. That is, order the fact clauses from most restrictive (matches fewest facts) to least restrictive. This can reduce the amount of memory used during rule evaluation. It may also improve the performance.

Sometimes these two guidelines conflict and it may require some experimentation to arrive at the best ordering.

### 12.2.6 Consider the Ordering of Tests in Rule Conditions

Similar to the recommendations for fact clauses, the tests in a rule condition should be ordered such that a test that will be more restrictive is placed before a test that is less

restrictive. This can reduce the amount of computation required for facts that do not satisfy the rule condition. If the degree of restrictiveness is not known, or estimated to be equal for a collection of tests, then the simpler tests should be placed before more expensive tests.

### 12.2.7 Use Functions Instead of AssertXPath and Supports XPath

Most of the work done by the rules engine is done during assert, retract, or modify operations. In particular, the `assertXPath` method, though very convenient, may have a performance impact. The power of this method is not only that it asserts the whole hierarchy in one call, but also asserts some XLink facts for children facts to link back to parent facts. However, if these features are not needed, and you need to assert only a few levels as facts, it is better to turn off the "Supports XPath" for the relevant fact types and then use a function to do custom asserts.

Instead of using assertXPath the following example uses a function to assert `ExpenseReport` and `ExpenseLineItems`:

```
function assertAllObjectsFromList(java.util.List objList)
{
 java.util.Iterator iter = objList.iterator();
 while (iter.hasNext())
 {
   assert(iter.next());
 }
}

function assertExpenseReport (demo.ExpenseReport expenseReport)
{
 assert(expenseReport);
 assertAllObjectsFromList(expenseReport.getExpenseLineItem());
}
```

To improve performance of assertXPath, select the "Enable improved assertXPath support for performance" check box in the Dictionary Properties page in Rule Author. Taking advantage of this will require that the following conditions are met:

- assertXPath is only invoked with an XPath expression of "//*". Any other XPath expression will result in an `RLIllegalArgumentException`.
- XLink facts should not be used in rule conditions as the XLink facts will not be asserted.

# 13

# Oracle BPEL Process Manager Performance Tuning

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

This chapter contains the following sections:



## 13.1 About BPEL Process Manager

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

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

## 13.2 Basic Tuning Considerations

This section describes the performance tuning properties at the BPEL engine level. They can be configured using Oracle Enterprise Manager. For more information , see "Configuring BPEL Process Service Engine Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

---

**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.

---

### 13.2.1 BPEL Threading Model

When the dispatcher must schedule a dispatch message for execution, it can enqueue the message into a thread pool. Each dispatch set can contain a thread pool

(java.util.concurrent.ThreadPoolExecutor). The BPEL thread pool implementation notifies the threads when a message has been enqueued and ensures the appropriate number of threads are instantiated in the pool.

---

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

---

#### 13.2.1.1 Dispatcher System Threads

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

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

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

#### 13.2.1.2 Dispatcher Invoke Threads

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

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

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

#### 13.2.1.3 Dispatcher Engine Threads

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

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

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

#### 13.2.1.4 Dispatcher Maximum Request Depth

The `dspMaxRequestDepth` property sets the maximum number of in-memory activities to process within the same request. After processing an activity request, Oracle BPEL Process Manager attempts to process as many subsequent activities as possible without jeopardizing the validity of the request. Once the activity processing

chain has reached this depth, the instance is dehydrated and the next activity is performed in a separate transaction.

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

The default value is 600 activities.

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

## 13.2.2 Tuning Audit Levels

The following properties can be set to audit levels.

### 13.2.2.1 AuditLevel

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

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

| Value | Description |
|---|---|
| Inherit | Inherits the audit level from infrastructure level. |
| Off | No audit events (activity execution information) are persisted and no logging is performed; this can result in a slight performance boost for processing instances. |
| Minimal | All events are logged; however, no audit details (variable content) are logged. |
| Error | Logs only serious problems that require immediate attention from the administrator and are not caused by a bug in the product. Using this level can help performance. |
| Production | All events are logged. The audit details for assign activities are not logged; the details for all other activities are logged. |
| Development | All events are logged; all audit details for all activities are logged. |

### 13.2.2.2 AuditDetailThreshold

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

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

The default value is 50000 (50 kilobytes).

#### 13.2.2.3 AuditStorePolicy

This property specifies the strategy to persist the BPEL audit data.

| Value | Description |
|---|---|
| syncSingleWrite (default) | AuditTrail and dehydration are persisted to DB in one transaction. |
| syncMultipleWrite | AuditTrail and dehydration are persisted in the same thread but separate transactions. |
| async | AuditTrail and dehydration are persisted by separate threads and separate transactions. |

By default, audit messages are stored as part of the main BPEL transaction. A BPEL instance holds on to the audit messages until the flow reaches dehydration. In some use cases, for example when you have a large loop, and there is no dehydration point in the loop, a large number of audit logs are accumulated. This could lead to an out-of-memory issue and BPEL main transaction can experience timeout errors. You may consider using `syncMultipleWrite` or `async` to store the audit message separately from the main transaction.

When you use `syncMultipleWrite` and `async` `auditStorePolicy`, there are a few other properties that need to be considered. Please see the sections below.

#### 13.2.2.4 AuditFlushByteThreshold

This property controls how often the engine should flush the audit events, basically after adding an event to the current batch, the engine checks to see if the current batch byte size is greater than this value or not.

Consider tuning this property when `async` or `syncMultipleWrite` audit strageties are used. This size needs to be tuned based on the application.

#### 13.2.2.5 AuditFlushEventThreshold

This property controls how often the engine should flush the audit events, basically when it reaches this limit of the number of events, the engine would trigger the store call.

Consider tuning this property when `async` or `syncMultipleWrite` audit strageties are used. This size needs to be tuned based on the application.

### 13.2.3 Tuning Database Persistence for BPEL

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

When setting the `oneWayDeliveryPolicy` property to async.cache, if the rate at which one-way messages arrive is much higher than the rate at which Oracle BPEL Server delivers them, or if the server fails, messages may be lost. In addition, the

system can become overloaded (messages become backlogged in the scheduled queue) and you may receive out-of-memory errors. Consult your own use case scenarios to determine if this setting is appropriate.

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

The `oneWayDeliveryPolicy` is from the Oracle 10*g* configuration property `deliveryPersistencePolicy`. The configuration property name in 11g is `bpel.config.oneWayDeliveryPolicy`.

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

### 13.2.4 Tuning Invoke Messages

The `MaximumNumberOfInvokeMessagesInCache` property specifies the number of invoke messages that can be kept in the in-memory cache. Once the engine hits this limit, the message is pushed to dispacther in-memory cache. The saved messages can be recovered using a recovery job. Use value -1 to disable.

The default value is 100000 messages.

### 13.2.5 Tuning Processed Requests List

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

This property is applicable to both durable and transient processes.

The default value is -1.

### 13.2.6 Tuning XML Document Persistence

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

This property is applicable to both durable and transient processes.

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

The default value is 10000 (100 kilobytes).

### 13.2.7 Validating XML

The `validateXML` property validates incoming and outgoing XML documents. If set to True, the Oracle BPEL Process Manager applies schema validation for incoming and outgoing XML documents. Nonschema-compliant payload data is intercepted and displayed as a fault.

This setting is independent of the SOA composite application and SOA Infrastructure payload validation level settings. If payload validation is enabled at both the service engine and SOA Infrastructure levels, data is checked twice: once when it enters the SOA Infrastructure, and again when it enters the service engine

CAUTION: Enabling XML payload validation can impact performance.

This property is applicable to both durable and transient processes.

The default value is False.

### 13.2.8 Tuning Wait Time

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

The default value is 45 seconds.

### 13.2.9 Tuning Instance Key Block Size

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

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

The default value is 10000.

### 13.2.10 Tuning Automatic Recovery Attempts

The `MaxRecoverAttempt` parameter allows you to configure the number of automatic recovery attempts to submit in the same recoverable instance. The value you provide specifies the maximum number of times invoke and callback messages are recovered. Once the number of recovery attempts on a message exceeds the specified value, a message is marked as nonrecoverable.

When a BPEL instance makes a call to another server using invokeMessage, and that call fails due to a server down, validation error, or security exception, the invokeMessage is placed in a recovery queue and BPEL attempts to retry those messages. When there are many messages, and a majority of them are being sent to the same target, the target can become overloaded. Setting the appropriate value of `MaxRecoveryAttempt` will prevent excessive load on servers that are targeted from BPEL web service calls.

## 13.3 Advanced Tuning Considerations

The following BPEL tuning considerations 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.

### 13.3.1 Tuning BPEL Properties Set Inside a Composite

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

All the properties set in this section affect the behavior of the component containing the BPEL process only. Each BPEL process can be created as a component of a composite. These properties can be modified in composite.xml or in the System MBean Browser of Oracle Enterprise Manager Fusion Middleware Control. For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

#### 13.3.1.1 Tuning Component Properties

The following component properties can be tuned for performance:

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

**Values:**

This property has the following values:

- False (default): instances are persisted completely and recorded in the dehydration store database.
- True: The completionPersist policy is used to determine persistence behavior. See Section 13.3.1.1.2.

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

| Value | Description |
|---|---|
| On (default) | The completed instance is saved normally |
| Deferred | The completed instance is saved, but with a different thread and in another transaction. |
| Faulted | Only the faulted instances are saved.<br>**Note:** When an unhandled fault occurs, regardless of these flags, audit information of the instance is persisted within `cube_instance` table. |
| Off | No instances of this process are saved. |

**13.3.1.1.3 auditLevel** You can set the audit level for a BPEL process service component. This setting takes precedence over audit level settings at the SOA Infrastructure, service engine, and SOA composite application levels.

Set the `bpel.config.auditLevel` property to an appropriate value in the composite.xml file of your SOA project as shown in the example below:

```
<component name="BPELProcess">
<implementation.bpel src="BPELProcess.bpel" />
<property name="bpel.config.auditLevel">Off</property>
</component>
```

| Value | Description |
|---|---|
| Inherit | Inherits the audit level from infrastructure level. |
| Off | No audit events (activity execution information) are persisted and no logging is performed; this can result in a slight performance boost for processing instances. |
| Minimal | All events are logged; however, no audit details (variable content) are logged. |
| Error | Logs only serious problems that require immediate attention from the administrator and are not caused by a bug in the product. Using this level can help performance. |
| Production | All events are logged. The audit details for assign activities are not logged; the details for all other activities are logged. |
| Development | All events are logged; all audit details for all activities are logged. |

### 13.3.1.2 Tuning Partner Link Properties

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

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

**Values:**

This property has the following values:

- False: Activity is dehydrated immediately after execution and recorded in the dehydration store. When `idempotent` is set to False, it provides better failover protection, but may impact performance if the BPEL process accesses the dehydration store frequently.
- True (default): If Oracle BPEL Server fails, it performs the activity again after restarting. This is because the server does not dehydrate immediately after the invoke and no record exists that the activity executed. Some examples of where this property can be set to True are: read-only services (for example, CreditRatingService) or local EJB/WSIF invocations that share the instance's transaction.

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

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

> **Note:** Invocations to the same partner link will happen in sequence and not in parallel. If you invoke different partner links each time with `nonBlockingInvoke` set to True, then each link will work in parallel even if all of the partner links point to the same source.

**Values:**

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

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

**Values:**

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

## 13.3.2 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. See Section 13.2.2, "Tuning Audit Levels" for audit configuration and Section 13.3.1.1.1, "inMemoryOptimization" for dehydration configuration. The table below describes the tables that are impacted by instance data growth. A brief description is provided of each table.

***Table 13–1 Oracle BPEL Process Manager Tables Impacted by Instance Data Growth***

| Table Name | Table Description |
|---|---|
| `audit_trail` | Stores the audit trail for instances. The audit trail viewed in Oracle BPEL Control is created from an XML document. As an instance is processed, each activity writes events to the audit trail as XML. |
| `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.<br>Audit details are separated from the `audit_trail` table due to their large size. If the size of a detail is larger than the value specified for this property, it is placed in this table. Otherwise, it is placed in the `audit_trail` table. |

*Table 13–1 (Cont.) Oracle BPEL Process Manager Tables Impacted by Instance Data*

| Table Name | Table Description |
|---|---|
| cube_instance | Stores process instance metadata (for example, the instance creation date, current state, title, and process identifier) |
| cube_scope | Stores the scope data for an instance (for example, all variables declared in the BPEL flow and some internal objects that help route logic throughout the flow). |
| dlv_message | Stores incoming (invocation) and callback messages upon receipt. This table only stores the metadata for a message (for example, current state, process identifier, and receive date). |
| dlv_subscription | Stores delivery subscriptions for an instance. Whenever an instance expects a message from a partner (for example, the receive or onMessage activity) a subscription is written out for that specific receive activity. |
| document_ci_ref | Stores cube instance references to data stored in the xml_ document table. |
| document_dlv_msg_ref | Stores references to dlv_message documents stored in the xml_document table. |
| 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, dlv_ message documents). This table stores the data as binary large objects (BLOBs). Separating the document storage from the metadata enables the metadata to change frequently without being impacted by the size of the documents. |
| Headers_properties | Stores headers and properties information. |

# 14

# Oracle Business Activity Monitoring Performance Tuning

This chapter describes how to tune the Oracle Business Activity Monitoring (BAM) dashboard application for optimal performance. Oracle BAM provides the tools for monitoring business services and processes in the enterprise.

This chapter discusses useful parameters that can be modified to enhance the overall performance of BAM:

- Section 14.1, "About Oracle Business Activity Monitoring"
- Section 14.2, "Basic Tuning Considerations"

## 14.1 About Oracle Business Activity Monitoring

Oracle Business Activity Monitoring (BAM) provides the tools for monitoring business services and processes in the enterprise. It allows correlating of market indicators to the actual business process and to changing business processes quickly or taking corrective actions if the business environment changes. Oracle BAM also provides the necessary tools and run-time services for creating dashboards that display real-time data inflow and define rules to send alerts under specified conditions.

For more information see *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*.

## 14.2 Basic Tuning Considerations

The following sections provide Oracle BAM tuning considerations that can be used to address performance issues:

- BAM Server Tuning
- BAM Dashboard Tuning
- BAM Database Tuning
- Internet Browser Tuning
- Enterprise Message Source Tuning

### 14.2.1 BAM Server Tuning

The following tuning configurations can be used to improve performance of the BAM Server:

#### 14.2.1.1 Set the ViewSetSharing and ElementCountLimit Parameters

The `ViewSetSharing` parameter can be set to `TRUE` or `FALSE` in the BAM server configuration file. This parameter enables view set sharing when possible. Typically a particular view set can be shared with other users if they are trying to access the same dashboard, if the view sets are not dissimilar due to factors like row level security or prompts/parameters tied to filters.

Consider setting the `ViewSetSharing` parameter to `TRUE` so that Active Data Cache (ADC) can reuse the same viewset and snapshot and avoid creating more viewsets. This reduces the BAM server resource usage and improves user response time.

If this parameter is turned on, it does not always guarantee that ADC can reuse the existing viewset. If there have been too many changes to the underlying snapshot for the existing viewset, ADC may choose to create new viewset instead.

The ReportCache parameter used to determine if there have been too many changes is `ElementsCountLimit`. This defines the number of changes to the snapshot used by Report Cache to do the determination. In cases where the active data comes in at a fast rate, try to set this parameter to a large number so that ADC can use view sharing at the expense of more server CPU usage. The default value of `ElementsCountLimit` is 50.

#### 14.2.1.2 Enable the Async Servlet

During periods of higher active data rates, the browser uses more memory. To prevent potential impacts to performance, consider providing more memory on the client machine. To do this, set the `UseAsynchServlet=TRUE` for the BAM dashboard application.

The BAM dashboard application uses the Async servlet feature so that the BAM server does not bind a specific thread to a specific user request. This provides for better server-side system resource usage.

This parameter can be turned off by adding `UseAsynchServlet=FALSE` in the server configuration file. During debugging, consider turning it off to make the process easier.

Otherwise this should always be turned on, which is the default.

See "Creating the Dashboard View" in *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*.

### 14.2.2 BAM Dashboard Tuning

This section provides information on tuning the BAM dashboard for performance.

#### 14.2.2.1 Tune the Active Data Retrieval Interval

The Active Data Retrieval Interval parameter controls the rate in milliseconds at which the Oracle BAM Active Data Cache (ADC) pushes events to the Oracle BAM Report Server. This is one of the factors that can affect the frequency of viewing active events on the dashboard page. Increasing this interval reduces the load on the Oracle BAM Server. Note that larger intervals increase the likelihood of multiple updates in the dashboard collapsing into a single update.

The default `ADCPushInterval` value is 1 second. You can override the default `ADCPushInterval` value within a particular report using the Active Data Retrieval Interval property in Active Studio.

For more information on using Active Studio, see "Getting Started With Oracle BAM Active Studio" in *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*.

### 14.2.3 BAM Database Tuning

To achieve the best performance for Oracle Business Activity Monitoring, consider maintaining a database on its own hardware dedicated to the Oracle Business Activity Monitoring system. General database administration practices, as described in the *Oracle Database Performance Tuning Guide*, also apply to a database dedicated to Oracle Business Activity Monitoring.

For more information on general database configurations, see Section 2.6, "Tuning Database Parameters".

### 14.2.4 Internet Browser Tuning

This section provides performance tuning configurations for Internet browsers:

#### 14.2.4.1 Set iActiveDataScriptsCleanupFactor

BAM sends active data in <script> blocks to the browser over a persistent connection. In some cases, the browser does not free up the memory used by the <script> blocks. This can impact dashboard performance over time.

The `iActiveDataScriptsCleanupFactor` parameter provides a solution for these memory issues. A periodic browser refresh is forced after receiving the specified number of characters. The issue may become apparent when active data is being sent to the dashboard at a fast pace. You may need to increase this value further for particularly high rates of data such as when active data is coming to the dashboard at a rate of 25 events per second or greater. Ultimately the value you set depends on factors like your data, number of views, number of viewsets, ADCPushinterval, and so on). You can monitor the browser's memory consumption to help determine an appropriate value.

If performance continues to be an issue, consider increasing the value for this parameter. For example, set the value to 2 or 3 times the default value if active data is predicted to increase. The default value for this parameter is 1048576 bytes. The default value often prevents frequent reconnects and prevents CPU/memory on the client machine from creeping up too high.

#### 14.2.4.2 Set Browser Cache Settings

If you are using Microsoft Internet Explorer, consider setting the Browsing History Settings to "Automatic." See the Microsoft Internet Explorer online help for more information.

### 14.2.5 Enterprise Message Source Tuning

BAM Enterprise Message Source (EMS) provides inbound JMS connectivity to BAM. After setup, a BAM EMS instance can monitor JMS queues/topics and read data from them. Each EMS instance is configured to publish data to a single Data Object in BAM Server. The Enterprise Message Source supports four types of operations: Insert, Update, Upsert, or Delete. Two types of JMS messages are supported: MapMessage and TextMessage.

#### 14.2.5.1 Message Batching

The EMS batching process clubs messages into one single message before it is sent to BAM EMS. This feature enables the sender to send all messages in one batch over JMS. The batching process can improve network performance by limiting the number of round trips from the sender to JMS server to BAM EMS.

# 15

# Oracle Mediator Performance Tuning

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



## 15.1 About Oracle Mediator

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

> **See Also:** For details about the SOA Suite, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
>
> For details about Oracle Mediator, see "Administering Oracle Mediator Service Components and Engines" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

## 15.2 Basic Tuning Considerations

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

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

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

### 15.2.1 Tuning metricsLevel

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

### 15.2.2 Using Domain-Value Maps

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

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

### 15.2.3 Deploying Deferred Routing Rules

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

- `DeferredWorkerThreadCount`: Specifies the number of deferred dispatchers for processing messages in parallel. For higher loads consider increasing this parameter to have more number of outbound threads for deferred processing as each parallel rule is processed by one of the `DeferredWorkerThreads`. Default value is 4 threads.
- `DeferredMaxRowsRetrieved`: When Mediator routing rule type is set to 'Parallel', `DeferredMaxRowsRetrieved` sets the number of maximum rows (maximum number of messages for parallel routing rule processing) that are retrieved from Mediator store table (which stores messages for parallel routing rule for processing.) Note that each message retrieved in this batch is processed by one worker thread at a time. Default value is 200 rows.
- `DeferredLockerThreadSleep`: For processing parallel routing rules, Oracle Mediator has a daemon locker thread that retrieves and locks messages from Mediator store database. The thread polls the database to retrieve messages for parallel processing. When no messages are available, the locker thread "sleeps" for the amount of time specified in the `DeferredLockerThreadSleep` and prevents round trips to database. Default value is 2 seconds. Consider increasing this value to improve performance. Some use case scenarios can benefit from a 'sleep' of 3600 seconds (60 minutes.)

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

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

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

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

### 15.2.4 Tuning Error and Retry Parameters

Consider increasing the `ErrorLockerThreadSleep` parameter value when you do not want to reduce the number of database trips.

The `ErrorLockerThreadSleep` parameter specifies the idle time between two successive iterations for retrieving errored out messages when there is no errored out message from parallel processing. The time is measured in seconds. Default value is 5 seconds. Consider increasing this value to improve performance. Some use case scenarios can benefit from an idle time of 3600 seconds (60 minutes.)

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

### 15.2.5 Setting the Audit Level

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

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

| Value | Description |
|---|---|
| Inherit | Inherits the audit level from infrastructure level. |
| Off | No audit events (flow execution information) are persisted and no logging is performed; this can result in a slight performance boost for processing instances. |
| Production | All events are logged. For each audit event, the payload details are not persisted. |
| Development | All audit events are logged. For each audit event, the payload details are also persisted. |

### 15.2.6 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 send them to the target services in an orderly manner.

For more information about Resequencers, refer to "Resequencing Messages" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

You can fine tune Resequencer by setting the value of the following properties in the Mediator Service Engine Properties page:

- `ResequencerWorkerThreadCount`: Specifies the worker thread count. Default is 4.
- `ResequencerMaxGroupsLocked`: Specifies the maximum number of groups locked in each iteration. Default is 4.
- `ResequencerLockerThreadSleep`: Specifies the sleep interval for the locker threads in seconds. Default is 10.

## 15.3 Tuning Event Delivery Network (EDN)

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

To improve performance of the Event Delivery Network, consider increasing the thread count (default is 3.) This property can be modified through WLST. For more information, see Section 11.3, "Modifying SOA Configuration Parameters".

# 16

# Oracle Business Process Management Performance Tuning

The Oracle Business Process Management (BPM) 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:



## 16.1 About Oracle Business Process Management

The Oracle BPM Suite provides an integrated environment for developing, administering, and using business applications centered around business processes. BPM 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 on using BPM, see the *Oracle Fusion Middleware User's Guide for Oracle Business Process Management*.

For more information on tuning Oracle BPM with your other Oracle Fusion Middleware components, see Chapter 2, "Top Performance Areas".

## 16.2 Basic Tuning Considerations

This section describes the following basic BPM performance tuning properties:

- Audit Level
- LargeDocumentThreshold
- Dispatcher System Threads
- Dispatcher Engine Threads
- Dispatcher Invoke Threads

**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.

### 16.2.1 Audit Level

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

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

| Value | Description |
|---|---|
| Inherit | Inherits the audit level from infrastructure level. |
| Off | No audit events (activity execution information) are persisted and no logging is performed; this can result in a slight performance boost for processing instances. |
| Minimal | All events are logged; however, no audit details (variable content) are logged. |
| Error | Logs only serious problems that require immediate attention from the administrator and are not caused by a bug in the product. Using this level can help performance. |
| Production | All events are logged. The audit details for assign activities are not logged; the details for all other activities are logged. |
| Development | All events are logged; all audit details for all activities are logged. |

### 16.2.2 LargeDocumentThreshold

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

This property is applicable to both durable and transient processes.

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

The default value is 10000 (100 kilobytes).

### 16.2.3 Dispatcher System Threads

The `dspSystemThreads` property specifies the total number of threads allocated to process system dispatcher messages. System dispatcher messages are general clean-up

tasks that are typically processed quickly by the server (for example, releasing stateful message beans back to the pool). Typically, only a small number of threads are required to handle the number of system dispatch messages generated during run time.

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

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

### 16.2.4 Dispatcher Engine Threads

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

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

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

### 16.2.5 Dispatcher Invoke Threads

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

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

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

## 16.3 Tuning Oracle Workspace and Worklist Applications

The following settings can be used to tune Oracle Workspace and Worklist applications:

| Parameter | Description |
|---|---|
| HTTP Session Timeout | To manage over resource usage, adjust the session timeout value, in minutes, in the web.xml file.<br>The following is a sample snippet of web.xml:<br>`<session-config>`<br>`<session-timeout>`<br>`5`<br>`</session-timeout>`<br>`</session-config>`<br>**NOTE**: If you must modify this property, post deployment, you must edit web.xml manually. See "Editing web.xml Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*. |
| ADF Client State Token | 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.<br>The following is a sample snippet of web.xml:<br>`<context-param>`<br>`<param-name>`<br>`org.apache.myfaces.trinidad.CLIENT_STATE_MAX_TOKENS`<br>`</param-name>`<br>`<param-value>`<br>`3`<br>`</param-value>`<br>`</context-param>`<br>**NOTE**: If you must modify this property, post deployment, you must edit web.xml manually. See "Editing web.xml Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*. |
| Compress_View_State Token | This setting controls whether or not the page state is compressed. Zipping greatly reduced the memory being taken up by page state in the session object.<br>The following is a snippet of the web.xml:<br>`<param-name>org.apache.myfaces.trinidad.COMPRESS_VIEW_STATE</param-name>`<br>`<param-value>`**`true`**`</param-value>` |
| DISABLE_CONTENT_COMPRESSION | By default, style classes that are rendered are compressed to reduce page size. In production environments, make sure you remove the DISABLE_CONTENT_COMPRESSION parameter from the web.xml file or set it to FALSE.<br>The following is a snippet of the web.xml:<br>`<param-name>org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION</param-name>`<br>`<param-value>`**`false`**`</param-value>` |

## 16.4 Tuning Process Analytics

Tuning Process Analytics includes the following:

### 16.4.1 Process Measurement

Process Analytics uses measurement events to sample the process and publish measurements to registered consumers. These measurements can be disabled using the BPMN Configuration "Disable Sensors". Specific consumers for these measurements can be disabled by setting the BPMN Configuration "Disable Actions". For more information, see the *Oracle Fusion Middleware Administrator's Guide*.

> **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 following, as needed, in a high volume environment:

- MeasurementTopic

  Bytes Max 800 MB

  Message Max 1000000

- MeasurementTopicConnectionFactory

  Send Timeout 240000

- BPMJMSServer

  MessageBuffer Size 100000

  Note that the BPMJMSServer uses a Paging File and JMSFileStore.

### 16.4.2 Tuning Process Cubes

Process Cubes perform periodic aggregations to compute workload information. The frequency of these computations is determined by the `CubeUpdateFrequency` parameter of `BPMNConfig` mbean and can be changed from the Oracle Enterprise Manager console. In a high volume environment, consider changing this parameter to an appropriately higher value such as 12 hours, for example, to conserve computing resources.

> **Note:** The creation of workload snapshots can impact performance. Consider using the properties in Oracle Fusion Middleaware Control to tune the frequency and time to live (TTL) for workload snapshots. For more information on using Fusion Middleware Contol, see the *Oracle Fusion Middleware Administrator's Guide*.

Process Cube Aggregator uses the `BPM_CUBE_AUDITINSTANCE` table to compute workload and performance information. Unwanted records from the `BPM_CUBE_AUDITINSTANCE` table get purged as part of the SOA Purge script. Additionally, consider running the following delete script periodically to purge the unwanted records from `BPM_CUBE_AUDITINSTANCE` table for improving the performance of Process Cube computations.

```
DELETE FROM BPM_CUBE_AUDITINSTANCE A
WHERE EXISTS
(SELECT 1 FROM BPM_CUBE_AUDITINSTANCE B
```

```
WHERE A.COMPONENTINSTANCEID = B.COMPONENTINSTANCEID AND
B.OPERATION='INSTANCE_CREATED' AND
B.ACTIVITYSTATUS='PROCESSED')
```

# 17

# Oracle Human Workflow Performance Tuning

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



## 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 *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

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

## 17.2 Monitoring Human Workflow Performance

By monitoring cycle time and other statistics, you can add staff to groups that are overloaded or take a longer time to complete. Thus reports can be used effectively to ensure workflows complete faster. By checking unattended tasks report, you can assign tasks that have been in the queue for a long time to specific users.

Several workflow reports (and corresponding views) are available that can make monitoring and proactively fixing problems easier. A few of these reports are listed below:

- The Unattended Tasks Report provides a list of group tasks that need attention since they have not yet been acquired by any user to work on.

- The Task Cycle Time Report gives an idea of how much time it takes for a particular type of workflow to complete.
- The Task Productivity Report indicates the inflow and outflow of tasks for different users.
- The Assignee Time Distribution Report provides a detailed drill-down of the time spent by each user during the task life cycle (including the idle time when the task was waiting to be picked up by a user.)

For more information on monitoring, see "Monitoring Oracle Fusion Middleware".

## 17.3 Basic Tuning Considerations

This section discusses the various options available to address performance issues:

- Minimizing Client Response Time
- Choosing the Right Workflow Service Client
- Narrowing Qualifying Tasks with Precise Filters
- Retrieving a Subset of Qualifying Tasks (Paging)
- Fetching Only the Information Needed for a Qualifying Task
- Reducing the Number of Return Query Columns
- Using the Aggregate API for Charting Task Statistics
- Using the Count API Methods for Counting the Number of Tasks
- Creating Indexes On Demand for Flexfields
- Using the doesTaskExist Method

### 17.3.1 Minimizing Client Response Time

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

### 17.3.2 Choosing the Right Workflow Service Client

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

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

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

remote method invocation (RMI) used in the remote client. Additional processing is performed by the web-services technology stack which causes the marshalling and unmarshalling of API method arguments between XML.

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

### 17.3.3 Narrowing Qualifying Tasks with Precise Filters

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

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

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

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

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

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

### 17.3.4 Retrieving a Subset of Qualifying Tasks (Paging)

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

For example, in the `queryTasks` method:

```
List tasks = querySvc.queryTasks(ctx,
```

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

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

### 17.3.5 Fetching Only the Information Needed for a Qualifying Task

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

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

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

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

Similarly, for attachments where the name of the attachment is to be displayed in the listing and the document itself is stored in an external repository, consider capturing the attachment name in the payload and mapping it to a flex field, so that processing time is optimized. While constructing the listing information, the link to the attachment can be constructed by fetching the appropriate flex field.

### 17.3.6 Reducing the Number of Return Query Columns

When using the `queryTask` service, consider reducing the number of query columns to improve the SQL time. Also, try to use the common columns as they are most likely indexed and the SQL can execute faster.

For example, in the following `queryTasks` method, only the `TASKNUMBER` and `TITLE` columns are returned:

```
List queryColumns = new ArrayList();
```

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

### 17.3.7 Using the Aggregate API for Charting Task Statistics

Sometimes it is necessary to display charts or statistics to summarize task information. Rather than fetching all the tasks using the query API, and computing the statistics at the client layer, consider using the new aggregate APIs to compute the statistics at the database level.

For example, the following call illustrates the use of the API to get summarized statistics based on state for tasks assigned to a user:

```
List taskCounts = querySvc.queryAggregatedTasks(ctx,
Column.getColumn(WFTaskConstants.STATE_COLUMN),
 ITaskQueryService.AssignmentFilter.MY,
keyWordFilter,
filterPredicate,
false,
false);
```

### 17.3.8 Using the Count API Methods for Counting the Number of Tasks

Sometimes it is only necessary to count how many tasks exist that match certain criteria. Rather than calling the `queryTasks` API method, and determining the size of the returned list, call the `countTasks` API method, which returns only the number of matching tasks. The performance impact of returning a count of tasks is much lower than returning a list of task objects.

For example, the following call illustrates the use of the API to get the total number of tasks assigned to a user:

```
int numberOfTasks = querySvc.countTasks(ctx,
ITaskQueryService.AssignmentFilter.MY,
keyWordFilter,
filterPredicate);
```

### 17.3.9 Creating Indexes On Demand for Flexfields

The workflow schema table WFTASK contains several flexfield attribute columns that can be used for storing task payload values in the workflow schema. Because there are numerous columns, and their use is optional, the installed schema does not contain indexes for these columns. In certain use-cases, for example, where certain mapped flexfield columns are frequently used in query predicates, performance can be improved if you create indexes on these columns.

For example, to create an index on the TEXTATTRIBUTE1 column, the following SQL command should be run:

```
create index WFTASKTEXTATTRIBUTE1_I on WFTASK(TEXTATTRIBUTE1);
```

**Note:** The exact indexes required depend on the flexfield attribute columns being used, and the nature of the queries being executed. After creating the indexes, the statistics for the WFTASK table should be re-computed and flushed.

### 17.3.10 Using the doesTaskExist Method

Sometimes it is necessary to check whether any tasks exist that match particular query criteria. Rather than calling the `countTasks` method, and checking if the number returned is zero, consider using doesTaskExist. The `doesTaskExist` method performs an optimized query that simply checks if any rows exist that match the specified criteria. This method may achieve better results than calling the `countTasks` method.

For example, the following call illustrates the use of the API method to determine if a user owns any task instances:

```
boolean userOwnsTask = querySvc.doesTaskExist(ctx,
ITaskQueryService.AssignmentFilter.OWNER,null,null);
```

## 17.4 Advanced Tuning Configurations

This section discusses the various options available to address complex performance issues:

- Improving Server Performance
- Completing Workflows Faster
- Tuning Identity Provider
- Tuning the Database

### 17.4.1 Improving Server Performance

Server performance essentially determines the scalability of the system under heavily loaded conditions. Section 17.3.1, "Minimizing Client Response Time" lists several ways in which client response times can be minimized by fetching the right of amount of information and reducing the potential performance impact associated with querying. These techniques also reduce the database and service logic performance impacts at the server and can improve server performance. In addition, a few other configuration changes can be made to improve server performance:

- Archive Completed Instances Periodically
- Select the Appropriate Workflow Callback Functionality
- Minimize Performance Impacts from Notification
- Deploy Clustered Nodes

#### 17.4.1.1 Archive Completed Instances Periodically

The database scalability of a system is largely dependent on the amount of data in the system. Since business processes and workflows are temporal in nature, once they are processed, they are not queried frequently. Having numerous completed instances in the system can slow the system. Consider using an archival scheme to periodically move completed instances to another system that can be used to query historical data. Archival should be done carefully to avoid orphan task instances.

#### 17.4.1.2 Select the Appropriate Workflow Callback Functionality

The workflow callback functionality can be used to query or update external systems after any significant workflow event, such as assignment or completion of task. While this functionality is very useful, it has to be implemented correctly to avoid impacting performance.

When performance is critical, ensure that there are sufficient resources to update the external system after the task is completed instead of after every workflow event. For example, instead of using a callback, the service can be invoked once after the completion of the task. If a callback cannot be avoided, then consider using a Java callback instead of a BPEL callback. Java callbacks do not have the performance impact associated with a BPEL callback since the callback method is executed in the same thread. In contrast, a BPEL callback may impact performance when sending a message to the BPEL engine, which in turn must be correlated so that it is delivered to the correct process instance. The workflow service has to be called by the BPEL engine after the invocation of the service.

#### 17.4.1.3 Minimize Performance Impacts from Notification

Notifications are useful for alerting users that they have a task to execute. In environments where most approvals happen through email, actionable notifications are especially useful. This also implies that there is not much load in terms of worklist usage. However if most users interact through the Worklist, and notifications serve a secondary purpose, then notifications should be used judiciously. Consider minimizing the notification to just alert a user when a task is assigned instead of sending out notifications for each workflow event. Also, if the task content is also mailed in the notification there may be an impact to performance. To minimize the impact, consider making the notifications secure in which case only a link to the task is sent in the notification and not the task content itself.

#### 17.4.1.4 Deploy Clustered Nodes

All workflow instances and state information are stored in the dehydration database. Workflow services are stateless which means they can be used concurrently on a cluster of nodes. When performance is critical and a highly scalable system is needed, a clustered environment can be used for supporting workflow. For more information on clustered architecture, see Section 33.2, "Using Clusters with Oracle Fusion Middleware".

### 17.4.2 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 improve the amount of time it takes to complete workflows. Some of these options are discussed in this section:

- Specifying Escalation Rules
- Using User and Group Rules for Automated Assignment
- Using Task Views to Prioritize Work

#### 17.4.2.1 Specifying 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.

#### 17.4.2.2 Using User and Group Rules for Automated Assignment

Instead of manually reassigning tasks to other users or members of a group, you can use user and group rules to perform automated reassignment. This ensures that workflows get timely attention. For example, a user can set up a user rule such that workflows of a specific type and matching a certain filter criteria are automatically reassigned to another user in a specified time window. Similarly, a group rule can be used to automatically reassign workflows to a member of the group based on different routing criteria such as round robin or most productive. Thus rules can help significantly reduce workflow waiting time, which results in faster workflow completion.

#### 17.4.2.3 Using Task Views to Prioritize Work

A user's inbox can contain tasks of various types with various due dates. The user has to manually sift through the tasks or sort them to find out which one he or she should work on next. Instead, by creating task views where tasks are filtered based on due dates or priority, users can get their work prioritized automatically so they can focus on completing their tasks instead of wasting their time on deciding which tasks to work on. This also results in faster completion of workflows.

### 17.4.3 Tuning Identity Provider

The workflow service uses information from the identity provider in constructing the SQL query to determine the tasks qualifying for a user based on his or her role/group membership. The identity provider is also queried for determining role information to determine privileges of a user when fetching the details of a task and determining what actions can the user perform on a task. There are a few ways to speed up requests made to the identity provider.

- Set the search base in the identity configuration file to node(s) as specific as possible. Ideally you should populate workflow-related groups under a single node to minimize traversal for search and lookup. This is not always possible; for example, you may need to use existing groups and grant membership to groups located in other nodes. If it is possible to specify filters that can narrow down the nodes to be searched, then you should specify them in the identity configuration file.
- Index all critical attributes such as dn and cn in the identity provider. This ensures that when a search or a lookup is done, only a subset of the nodes are traversed instead of a full tree traversal.
- Use an identity provider that supports caching. Not all LDAP providers support caching but Oracle Internet Directory supports caching which can make lookup and search queries faster.

### 17.4.4 Tuning the Database

The Human Workflow schema is shipped with several indexes defined on the most important columns for all the tables. Based on the type of request, different SQL queries are generated to fetch the task list for a user. The database optimizer evaluates the cost of different plan alternatives (for example, full table scan, access table by index) and decides on a plan that is lower in cost. For the optimizer to work correctly, the index statistics should be current at all times. As with any database usage, it is important to make sure the database statistics are updated at regular intervals and

other tunable parameters such as memory, table space, and partitions are used effectively to get maximum performance.

For more information on tuning the database, see Section 2.6, "Tuning Database Parameters".

# 18

# Oracle Adapters Performance Tuning

This chapter describes how to tune Oracle Adapters for optimal performance. Oracle Adapters, a component of the Oracle SOA Suite of Applications, provide an integrated view of data and allow multiple applications to be integrated.

This chapter contains the following sections:



## 18.1 About Oracle Adapters

Oracle technology adapters integrate Oracle Application Server and Oracle Fusion Middleware components such as Oracle BPEL Process Manager (Oracle BPEL PM) or Oracle Mediator components to file systems, FTP servers, database queues (advanced queues, or AQ), Java Message Services (JMS), database tables, and message queues (MQ Series).

For more information on Oracle Adapters, see *Oracle Fusion Middleware User's Guide for Technology Adapters*.

## 18.2 Oracle JCA Adapters for Files/FTP

This section describes the various features available for scalability and performance tuning of Oracle File and FTP Adapters.The Oracle File and FTP Adapters provide knobs to throttle the inbound and outbound operations. The Oracle File and FTP Adapters also provide knobs that can be used to tune the performance of outbound operations. The Oracle File and FTP Adapters knobs are described in the following sections:

**Note:** For composites with Oracle File and FTP Adapters, which are designed to consume very large number of concurrent messages, you must set the number of open files parameter for your operating system to a larger value. For example, to set the number of open files parameter to 8192 for Linux, use the `ulimit -n` 8192 command

### 18.2.1 Inbound Throttling Best Practices

The Oracle File and FTP Adapters provide parameters that can be used to throttle the inbound operations. The table below describes the inbound throttling practices:

| Parameter | Type | Values | Description |
|---|---|---|---|
| `MaxRaiseSize` | JCA | `<property name="MaxRaiseSize" value="100"/>`<br>Default: 10000 (ten thousand) | This parameter defines the maximum number of files that the inbound adapter would submit for processing on each polling cycle. For example, if your inbound directory has 1000 files and the `MaxRaiseSize` is set to `100`, the adapter can increase to 100 files on each polling cycle.<br>Defined in the Inbound JCA File. |
| `SingleThreadModel` | JCA | `<property name="SingleThreadModel" value="true"/>`<br>Default: False (In this case, the global in-memory queue is used). | If the value is `true`, the poller lists, translates, or publishes files in the same thread. In other words, it does not use the global in-memory queue for publishing.<br>Defined in the Inbound JCA File. |
| `ThreadCount` | JCA | `<property name="ThreadCount" value="10"/>`<br>Default: -1 (In this case, the adapter uses the global thread pool and in-memory queue) | This parameter enables the Oracle File and FTP Adapters to create their own processor threads rather than depending on the global pool of processor worker threads for processing the enqueued files. This parameter partitions the in-memory queue and each composite application receives its own in-memory queue.<br>If the `ThreadCount` is set to `0`, then the threading behavior is the same as that of the SingleThreadModel. If the `ThreadCount` is set to `-1`, then the global thread pool is activated, which is the same as the Default Threading Model. The maximum value that can be set for `ThreadCount` is `40`.<br>Defined in the Inbound JCA File. |

### 18.2.2 Outbound Throttling Best Practices

The Oracle File and FTP Adapters provide parameters that can be used to throttle the outbound operations. The table below describes the outbound throttling practices:

| Parameter | Type | Value | Description |
|---|---|---|---|
| ConcurrentThreshold | JCA | <property name="ConcurrentThreshold" value="100"/><br>Default: 20 (In this case, not more than 20 translations occur for a particular outbound scenario.) | This parameter specifies the maximum number of translation activities that are allowed to start in parallel for a particular outbound scenario. The translation step during the outbound operation is CPU intensive and must be monitored as it might cause other applications or threads to starve. The maximum value is 100.<br>Defined in the Outbound JCA File. |

### 18.2.3 Outbound Performance Best Practices

The Oracle File and FTP Adapters provide parameters that can be used to tune the performance of outbound operations. The table below describes the outbound performance parameters:

| Parameter | Type | Value | Description |
|---|---|---|---|
| UseStaging | JCA | <property name="UseStaging" value="true"/><br>Default: True | If the parameter is set to true, then the outbound Oracle File or FTP Adapter writes translated data to a staging file and later streams the staging file to the target file. If the parameter is set to false, then the outbound Oracle File or FTP Adapter does not use an intermediate staging file.<br>Defined in Outbound JCA File. |
| serializeTranslation | Endpoint Property | <reference name="PurchaseOrderOut"><br><interface.wsdl interface="...."/><br><binding.jca config="PurchaseOrderOut_ftp.jca"/><br><property name="serializeTranslation" type="xs:string" many="false" source="" override="may">true</property><br></reference><br>Defaults:<br>■ True (If the value of UseStaging is set to True)<br>■ False (If the value of UseStaging is set to False) | If True, then the translation step is serialized using a semaphore. The number of permits for semaphore (monitoring the translation step) comes from ConcurrentThreshold parameter (listed in the preceding table). The default value of True is used because the translation step is CPU intensive and you do not want to starve other applications or threads.<br>If False, then the translation step occurs outside the semaphore.<br>Defined in Binding property for reference in composite.xml. |

| Parameter | Type | Value | Description |
|---|---|---|---|
| inMemoryTranslation | Binding Property | <reference name="PurchaseOrderOut"> <interface.wsdl interface="...."/> <binding.jca config="PurchaseOrderOut_ftp.jca"/> <property name="inMemoryTranslation" type="xs:string" many="false" source=""override="may">false</property> </reference><br>Default: False | This parameter is applicable only if UseStaging is False.<br>If True, then the translation step occurs in-memory (an in-memory byte array is created.)<br>If False, then the adapter creates an output stream to the target file (FTP, FTPS, and SFTP included) and allows the translator to translate and write directly to the stream.<br>Defined in Binding property for reference in composite.xml. |

# 18.3 Oracle JCA Adapter for Database Tuning

The Oracle Database Adapter is pre-configured with many performance optimizations. You can, however, make some changes to reduce the number of round trips to the database, as described in the following sections:

- JCA Adapter Basic Tuning Considerations
- Existence Checking

> **Note:** The tuning considerations in this chapter are listed for example only. Tuning parameters are specific to each deployment. Review you current usage and performance issues to determine which tuning considerations can improve performance.

## 18.3.1 JCA Adapter Basic Tuning Considerations

Adapter performance is directly related to the number of round-trips to the database, and the network cost of each trip. If performance becomes an issue, and making modifications is appropriate for your deployment, consider tuning the following parameters:

- Use Indexes

  Indexes can improve performance of selects, updates and deletes. Index all queried fields, such as the primary key and the MarkReadField of the LogicalDeletePollingStrategy, when polling. For MarkReadField specify a non-null MarkUnreadValue. Caution: An index on a column containing many nulls may revert to full table scans.

- Disable DetectOmissions

  The DetectOmissions parameter allows the detection of XML elements for which no value was specified. The related columns are excluded from inserts and updates. Disabling this parameter generally improves performance, but there is one case where it could have a negative effect. If multiple rows are being passed in as a single XML, and each row has different columns set (user entered with many optional fields), there is no benefit from batch writing, as each insert or update is different.

- Increase `MaxRaiseSize`

  The `MaxRaiseSize` parameter indicates the maximum number of XML records that can be raised at a time to the BPEL engine. For example, if you set `MaxRaiseSize` = 10, then 10 database records are raised simultaneously. On an inbound read, for example, you can set `MaxRaiseSize` = 0 (unbounded) which means that if you read 1000 rows, you can create one XML with 1000 elements. These elements are passed through a single Oracle BPEL Process Manager instance. A merge on the outbound side can then take all 1000 in one group and write them all at once with batch writing. Use the `MaxRaiseSize` parameter for publishing large payloads.

- Increase `MaxTransactionSize`

  This property controls the number of records processed per transaction by each thread. If set to a large value such as 1000, turning on the `UseBatchDestroy` option could have a negative impact on performance. Setting a large `MaxTransactionSize` and a small `MaxRaiseSize` could also have negative impact on performance. Consider maintaining up to a 10:1 ratio in a synchronous scenario. Ideally, you should consider increasing `MaxRaiseSize` until it is a 1:1 ratio.

- Enable `UseBatchDestroy`

  This property controls how the processed records are updated (ex: `Deleted` for `DeletePollingStrategy`, `MarkedProcessed` for `LogicalDeleteStrategy`). If set, only one update/delete is executed for all the rows that are part of that transaction. The number of rows in a transaction is controlled by the `MaxTransactionSize` option. Note that this may not always offer an improvement because, by default, batch writing is used, which also ends up in a single round trip to the database.

- Enable `Batch Reading`

  Batch reading of one-to-many and one-to-one relationships is on by default. You can also use joined reading for one-to-one relationships instead, which may offer a slight improvement.

- Disable `Delete Polling Strategy`

  Avoid the delete polling strategy because it must individually delete each row. The sequencing polling strategy can destroy 1000 rows with a single update to a helper table. Note that a `LogicalDelete` is also better than Delete, as updates are typically faster than deletes. To maintain performance, however, ensure that you have indexed the table. If you have not indexed, you can keep the total number of rows small by using deletes. In some instances deletes may be faster as the cost of a full table scan is negligible.

- Use `Distributed Polling`

  Distributed polling enables you to configure polling for scalability. For more information, see "Scalability" in *Oracle Fusion Middleware User's Guide for Technology Adapters*.

- Use `Synchronous Processes`

  On BPEL you can configure Database Adapter processes to be synchronous. You can also create sequential routing rules in Mediator. This can improve throughput in database-to-database scenarios, as there is less instance processing impact.

- Use `Insert`

The insert operation is the most performant because it uses no existence check and has no extra performance impact associated with it. There are no reads, only writes. If you know that you are inserting most of the time, use insert, and catch a unique key constraint SQL exception inside your BPEL process, which can then perform a merge or update instead.

To monitor performance, you can enable debug logging and then watch the SQL for various inputs.

- Disable `Merge`

  Merge executes one extra SELECT per related table. The SELECT is used to determine whether each row should be inserted or updated. If the row is updated, the update performed is minimal. If no rows have changed, nothing is updated.

- Use `Connection Pooling`

  The adapter should also point to a tuned data source connection pool. Tuning the connection pool is important because creating and tearing down database connections can impact performance.

- Use `Attribute Filtering`

  On the Attribute Filtering page of the Adapter Configuration Wizard you can choose which fields to map to the XML and vice versa. You can improve performance by deselecting columns that are not needed for your particular business case, especially large columns like LOBs.

- Use `Native Sequencing`

  If you are using the XSL functions to assign primary keys to records, consider using the built-in native sequencing support in the adapter. Sequencing support obtains and caches 50 keys at a time by default. Caching improves performance by reducing the number of round trips. The chunk size can be controlled incrementally by modifying the `sequencePreallocationSize` connector property.

- Do not use primary or foreign keys on the database

  Using primary and foreign keys can impact performance. Avoid using them when possible.

- JDBC Driver Class

  The default JDBC driver class used to create the physical database connections in the connection pool is `oracle.jdbc.xa.client.OracleXADataSource`. Changing the driver to `oracle.jdbc.OracleDriver` may provide some performance improvement.

  For more information on tuning the JDBC drivers, see "Third Party JDBC Driver and Database Connection Configuration" in *Oracle Fusion Middleware User's Guide for Technology Adapters*.

## 18.3.2 Existence Checking

One method of performance optimization for `merge` is to eliminate check database existence checking. The existence check is marginally better if the row is new, because only the primary key is returned, not the entire row. Due to the nature of merge, however, if the existence check passes, the entire row must be read to calculate what changed. Therefore, for every row to be updated, you see one extra round trip to the database during merge.

Use check cache on the root descriptor/table and any child tables if A is master and B is a privately owned child. If A does not exist, B cannot exist. And if A exists, all of its child tables are loaded as part of reading A.

---

**Note:** One way to prevent merge from performing an existence check for every record, when you know that an insert is required, is to set the primary key to null.

---

### 18.3.3 Throttling

It is possible to configure a speed limit on DbAdapter performance to protect down-stream components from message bursts. Consider leaving burst records unprocessed on the source database until SOA can process them efficiently. As of Oracle Adapters release 11.1.1.6.0 you can set the inbound DbAdapter property `RowsPerPollingInterval`. It acts as a limit on the number of records which can be processed in one polling interval. The default is unlimited.

The following sections describe the configuration options for `RowsPerPollingInterval`:

#### 18.3.3.1 Formula

The formula for maximum rows per second is:

Number of active nodes in SOA cluster x NumberOfThreads x RowsPerPollingInterval / PollingInterval

#### 18.3.3.2 RowsPerPollingInterval and MaxTransactionSize

`MaxTransactionSize` can be thought of as `RowsPerDatabaseTransaction` or `DatabaseFetchSize`. It does not affect how many rows can be processed in one polling interval period.

The one exception is the following configuration:

```
-distributed polling checked, usesSkipLocking="false"
```

In this one case `RowsPerPollingInterval` will default to `MaxTransactionSize` instead of unlimited.

If `RowsPerPollingInterval` is set to lower than `MaxTransactionSize` or `MaxRaiseSize`, they will be effectively lowered to `RowsPerPollingInterval`.

#### 18.3.3.3 Configuration

There is no UI support for `RowsPerPollingInterval`. Instead find the db.jca file for the inbound polling service and add the property manually. Add it to the same section as the properties `MaxRaiseSize`, `MaxTransactionSize`, and `PollingInterval`, in any order.

## 18.4 Oracle Socket Adapter Tuning

This section describes performance tuning for Oracle Socket Adapter. Performance can be optimized for the Oracle Socket Adapter using Connection Pool if the socket server you are connecting to does not close the socket with each interaction. Connection pool lets you use a socket connection repeatedly, avoiding the overload of creating a new socket for each interaction.

**Note:** The Connection Pool feature is applicable to outbound interactions only. For more information on Socket Adapters, see "Oracle JCA Adapter for Sockets" in *Oracle Fusion Middleware User's Guide for Technology Adapters*

In order to enable the connection pool feature for the Oracle Socket Adapter, the `KeepAlive` connection factory property must be set to `True`. This connection property can be modified using the Connection Pool tab of Oracle WebLogic Server Administration Console.

For instructions on modifying the Oracle Socket Adapter connection pooling, see "Configuring Oracle Socket Adapter Connection Pooling" in *Oracle Fusion Middleware User's Guide for Technology Adapters*.

## 18.5 Oracle SOA JMS Adapter Tuning

This section describes some of the properties that can be set for the Oracle SOA JMS Adapter to optimize performance. See "Introduction to the Oracle JMS Adapter" in the *Oracle Fusion Middleware User's Guide for Technology Adapters* for more information.

### 18.5.1 adapter.jms.receive.threads Property

To improve performance, the `adapter.jms.receive.threads` property can be tuned for an adapter service. The default value is 1, but multiple inbound threads can be used to improve performance. When specified, the value of `adapter.jms.receive.threads` is used to spawn multiple inbound poller threads.

For example:

```
<service name="dequeue" ui:wsdlLocation="dequeue.wsdl">
<interface.wsdl
interface="http://xmlns.oracle.com/pcbpel/adapter/jms/textmessageusingqueues/textm
essageusingqueues/dequeue%2F#wsdl.interface(Consume_Message_ptt)"/>
<binding.jca config="dequeue_jms.jca">
<property name="adapter.jms.receive.threads" type="xs:string"
many="false">10</property>
</binding.jca">
</service>
```

## 18.6 Oracle AQ Adapter Tuning

This section describes Oracle AQ Adapter tuning configurations.

### 18.6.1 adapter.aq.dequeue.threads Property

To improve dequeue performance 'adapter.aq.dequeue.threads' property can be set for an adapter service. Default value is 1 but multiple inbound threads can be used to improve performance. The value of property 'adapter.aq.dequeue.threads' is used to spawn multiple inbound poller threads.

For example:

```
<service name="dequeue" ui:wsdlLocation="dequeue.wsdl">
<interface.wsdl
interface="http://xmlns.oracle.com/pcbpel/adapter/aq/raw/raw/dequeue/#wsdl.interfa
ce(Dequeue_ptt)"/>
```

```
<binding.jca config="dequeue_aq.jca">
<property name="adapter.aq.dequeue.threads" type="xs:string"
many="false">10</property>
</binding.jca>
</service>
```

## 18.7 Oracle MQ Adapter Tuning

The Oracle MQ Series Adapter supports the scalability feature for inbound operations only. Oracle MQ Series Adapter provides the parameter to control the number of threads that dequeue the messages from the inbound queue.You must specify the following property in the.jca file:

```
InboundThreadCount='N'
```

In the example above *N* is the number of threads that you want to span to dequeue the messages from the inbound queue.

# 19

# User Messaging Service Performance Tuning

This chapter describes tips for tuning the User Messaging Service. It contains the following sections:



## 19.1 About Oracle User Messaging Services

Oracle User Messaging Service enables users to receive notifications sent from SOA applications that are developed and deployed to the Oracle WebLogic Server using Oracle JDeveloper.

At the application level, there is notification activity for a specific delivery channel (such as SMS or E-Mail). For example, when you build a SOA application that sends e-mail notification, you drag and drop an Email Activity component from the JDeveloper Component Palette to the appropriate location within a workflow. The application connects then sends notifications.

For more information on Oracle User Messaging Service, see *Oracle WebLogic Communication Services Administrator's Guide*, *Oracle WebLogic Communication Services Developer's Guide*, and the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

## 19.2 Basic Tuning Considerations

Depending on your User Messaging usage and performance issues, you may consider tuning the following:

- SMPP Driver Performance Tuning
- Email Driver Polling Frequency

### 19.2.1 SMPP Driver Performance Tuning

Short Messaging Peer-Peer Protocol (SMPP) messaging drivers can be configured using Enterprise Manager. One of the key parameters for optimizing SMPP performance is `WindowSize`. This is especially important when the SMPP driver is connected to a remote SMSC and there is high network latency between the two elements. Configuring the `WindowSize` parameter enables the SMPP driver to send several requests to the Short Messaging Service Center (SMSC) before waiting for an

acknowledgment. Without windowing (i.e., a WindowSize of 1), the driver must wait for a synchronous acknowledgment from the SMSC before sending the next message. With windowing, more messages can be sent per network round-trip, allowing a higher overall throughput.

To take advantage of an increased `WindowSize`, the number of MDB threads for the driver must be correspondingly increased. The two values should be matched so that driver threads can process and send messages before waiting for the requests to be acknowledged. Increasing the two values may improve performance, but only up to the point at which network latency no longer dominates the sending rate. Also, the maximum allowed value for the `WindowSize` is normally defined as a service policy by the SMSC operator.

For more information, see "Configuring Oracle User Messaging Service" in *Oracle WebLogic Communication Services Administrator's Guide*.

### 19.2.2 Email Driver Polling Frequency

For Email drivers, the "`CheckMailFreq`" configuration parameter defines how frequently the driver checks for incoming emails. For example, a value of "30" means the driver checks the configured inbox every 30 seconds. This parameter can influence performance; checking more frequently enables the driver to keep up with a higher incoming email load, but can impact performance due to frequent IMAP or POP3 operations. Default value is 30 seconds.

## 19.3 Database Tuning for Optimal Throughput

User Messaging Service stores messaging state such as sent and received messages and delivery status information in the database. Therefore, database and data source tuning may have an effect on messaging throughput. The connection pool size for the data sources can be tuned for higher load levels, but the defaults are sufficient for most cases.

For general database tuning considerations, see Section 2.6, "Tuning Database Parameters".

# 20

# Oracle B2B Performance Tuning

This chapter describes tips for tuning Oracle B2B performance. It contains the following sections:



## 20.1 About Oracle B2B

Oracle B2B (Business to Business) is an e-commerce gateway that enables the secure and reliable exchange of business documents between an enterprise and its trading partners. Oracle B2B supports business-to-business document standards, security, transports, messaging services, and trading partner management. With Oracle B2B used as a binding component within an Oracle SOA Suite composite application, end-to-end business processes can be implemented.

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

## 20.2 Basic Tuning Considerations

The following sections describe basic tuning configurations that you should also consider while tuning:



### 20.2.1 Tuning MDS Cache Size

Changing the value of the Metadata Service (MDS) instance cache size can improve performance. A ratio of 5:1 is recommended for the `xmx-to-mdsCache` values. For example, if the `xmx` size is 1024, maintain `mdsCache` at 200 MB.

These settings can be modified using Oracle Enterprise Manager Fusion Middleware Control. For more information, see "Configuring Oracle B2B" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

### 20.2.2 Tuning Number of Threads

Changing the value of `b2b.inboundThreadCount` and `b2b.outboundThreadCount` can improve Oracle B2B message processing. The

recommended value depends on your system. For a 2 GB computer, for example, a setting of 3 to 5 is recommended. The `b2b.inboundThreadSleepTime` and `b2b.outboundThreadSleepTime` properties put a thread to sleep after message processing. A setting between 10 and 1000 (milliseconds) is recommended.

These settings can be modified using Oracle Enterprise Manager Fusion Middleware Control. For more information, see "Configuring Oracle B2B" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

### 20.2.3 Tuning the JMS Multiple Out Queues Setting

The JMS Out Queue component is the element that enables B2B to receive data from a JMS queue. To maximize performance, consider enabling the Multiple JMSOUTQUEUES and create the corresponding listening channels in B2B.

# 21

# Oracle Service Bus Performance Tuning

This chapter describes tips for tuning Oracle Service Bus performance. It contains the following sections:



## 21.1 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 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 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 routing the response back to the client.

It is necessary, therefore, 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*.

## 21.2 Basic Tuning Considerations

Depending on your OSB usage and performance issues, you may consider tuning the following:

- WebLogic Server Tuning

## 21.2.1 JVM Memory Tuning

JVM parameters can have an impact on OSB performance. The two primary JVM tuning parameters to consider when optimizing OSB performance are heap size and garbage collection. For more information on tuning the JVM for performance, see Section 2.4, "Tuning Java Virtual Machines (JVMs)".

## 21.2.2 WebLogic Server Tuning

To optimize OSB, consider tuning the following WebLogic Server parameters:

### 21.2.2.1 Domain Mode

For production environments, create a domain in "Production" mode to maximize performance. The parameter is:

```
-Dweblogic.ProductionModeEnabled=true
```

To enable Weblogic server production mode through Weblogic Administration Console, see *Oracle Fusion Middleware Understanding Domain Configuration for Oracle WebLogic Server*.

### 21.2.2.2 WebLogic Server Logging Levels

For OSB performance testing and production environments, consider using the lowest acceptable logging level, such as "ERROR" or "WARNING" whenever possible. For more information, see Section 2.10, "Setting Logging Levels"

### 21.2.2.3 HTTP Access Logging

To optimize OSB perfomance, consider turning off the HTTP access logging. For more information, see Section 6.3.3.1, "Access Logging".

### 21.2.2.4 JMS Tuning

Ensure that the right persistence level is set for the Java Message Service (JMS) destinations. Consider the following scenarios:

- For non-persistent JMS scenarios:

  Explicitly turn off persistence at the JMS server level by un-checking the "Store Enabled" flag from the Advanced section of the General tab for the JMS server on the WebLogic Server console. It is also possible to override the persistence mode at the JMS destination level.

- For persistent JMS scenarios:

  There are two choices: file store and JDBC store. 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 more information on JMS Server Tunings, see "Tuning WebLogic JMS" in the *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

### 21.2.2.5 Connection Backlog Buffering

You can tune the number of connection requests that a WebLogic Server instance will accept before refusing additional requests. The Accept Backlog parameter specifies how many Transmission Control Protocol (TCP) connections can be buffered in a wait

queue. This fixed-size queue is populated with requests for connections that the TCP stack has received, but the application has not accepted yet. This parameter should be tuned when dealing a large number of concurrent clients. For more information, see "Tuning Connection Backlog Buffering" in *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

## 21.3 Tuning OSB Operational Settings

This section discusses the following Oracle Service Bus operational settings:

- OSB Monitoring
- OSB Tracing
- Cache Tuning for Proxy Service Run-Time Data

### 21.3.1 OSB Monitoring

Though the out-of-the-box monitoring sub-system has a very low overhead and scales well to a large number of services as well as to multiple nodes in a cluster, when dealing with thousands of services or a large scale cluster deployment, being selective about enabling monitoring can help reduce network traffic. When a business or proxy service is created, monitoring is disabled by default for that particular service. For more information, see "Configuring Operational Settings for Proxy Services" or "Configuring Operational Settings for Business Services" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

To enable or disable monitoring of all services that have individually been enabled or disabled for monitoring, use the "Enable Monitoring" option on the **Operations Global Settings** page. For more information, see "Enabling Global Settings" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

### 21.3.2 OSB Tracing

Oracle Service Bus has the option to trace messages without having to shutdown the server. 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.

Tracing is disabled by default but can be enabled on a per service basis. When tracing is enabled, the entire message context is also printed including headers and message body. It is important to realize its impact for large message sizes and high throughput scenarios.

For more information, see "How to Enable or Disable Tracing" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

### 21.3.3 Cache Tuning for Proxy Service Run-Time Data

OSB caches proxy service runtime meta-data using a two-level cache with static and dynamic sections. The cache introduces a performance tradeoff between memory consumption and compilation cost. Note that caching proxy services may help throughput but could impact memory usage.

The static section is an upper-bound Least Recently Used (LRU) cache that is never garbage collected. When a proxy service is bumped from the static section, it is demoted to the dynamic section where the cache can be garbage collected when there is memory pressure.

The number of proxy services in the static portion of the cache can be tuned by setting its size using the system property `com.bea.wli.sb.pipeline.RouterRuntimeCache.size`. The default value is 100. This can be increased to a desired value provided there is sufficient memory for runtime data processing for large number of proxy services.

This property value can be set in the setDomainEnv.sh file as an extra java argument as follows:

```
-Dcom.bea.wli.sb.pipeline.RouterRuntimeCache.size={size}
```

Example:

```
EXTRA_JAVA_PROPERTIES="-Dcom.bea.wli.sb.pipeline.RouterRuntimeCache.size=3000
${EXTRA_JAVA_PROPERTIES}"
```

## 21.4 Transport Tuning (Oracle WebLogic Server and Oracle Service Bus)

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:

### 21.4.1 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 *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* |
| FTP Transports | 60 seconds | " FTP Transport Configuration Page" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* |
| MQ Transport | 1000 milliseconds | "MQ Transport Configuration Page" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* |
| SFTP Transport | 60 seconds | "SFTP Transport Configuration Page" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* |
| JCA Transport | 60 seconds | "JCA Transport Configuration Page" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*<br>See also Section 18.3.1, "JCA Adapter Basic Tuning Considerations" |

### 21.4.2 Read Limit

The read limit determines the number of files or messages that are read per polling sweep. This defaults to 10 for the File and FTP transports. It can be set to 0 to specify no limit. Set this value to the desired concurrency. For more information, see " File Transport Configuration Page" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

> **Note:** 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 OOM (out-of-memory error) if the message size is large.

## 21.5 Design Time Considerations for Proxy Applications

Consider the following design configurations for proxy applications based on your OSB usage and use case scenarios:

- 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.

- Transforming contents of a context variable such as `$body`.

  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.

- Use `$body/*[1]` to represent the contents of `$body` as an input to a Transformation (XQuery / XSLT) resource.

  OSB treats "`$body/*[1]`" as a special XPath that can be evaluated without invoking the XQuery engine. This is faster than specifying an absolute path pointing to the child of `$body`. A general XPath like "`$body/Order`" must be evaluated by the XQuery engine before the primary transformation resource is executed.

- Enable Streaming for pure Content-Based Routing scenarios.

  Read-only scenarios such as Content-Based Routing can derive better performance from enabling streaming. OSB leverages the partial parsing capabilities of the XQuery engine when streaming is used in conjunction with indexed XPaths. Thus, the payload is parsed and processed only to the field referred to in the XPath. Other than partial parsing, an additional benefit for read-only scenarios is that streaming eliminates the overhead associated with parsing and serialization of XmlBeans.

  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. See Section 21.6, "Design Considerations for XQuery Tuning" for additional details.

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.

  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.

  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.

- 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.

## 21.6 Design Considerations for XQuery Tuning

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>
```

- Avoid the use of double front slashes ("//") in XPaths.

  $body//CustName while returning the same value as $body/Order/CtrlArea/CustName will perform a lot worse than the latter expression. "//" 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 XPaths where applicable.

  An XPath can be indexed by simply adding "[1]" after each node of the path. XQuery is a declarative language and an XPath can return more than one node; it can return an array of nodes. `$body/Order/CtrlArea/CustName` implies

returning all instances `Order` under `$body` and all instances of `CtrlArea` under `Order`. Therefore, the entire document has to be read in order to correctly process the above XPath. If you know that there is a single instance of `Order` under `$body` and a single instance of `CtrlArea` under `Order`, we could rewrite the above XPath as `$body/Order[1]/CtrlArea[1]/CustName[1]`.

The second XPath implies returning the first instances of the child nodes. Thus, only the top part of the document needs to be processed by the XQuery engine resulting in better performance. Indexing is key to processing only what is needed.

---

**Note:** Indexing should not be used when the expected return value is an array of nodes. For example, `$body/Order[1]/ItemList[1]/Item` returns all "Item" nodes, but `$body/Order[1]/ItemList[1]/Item[1]` only returns the first item node. Another example is an XPath used to split a document in a "for" action.

---

- 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. Sample XPaths with common context:

  ```
  $body/Order[1]/Summary[1]/Total, $body/Order[1]/Summary[1]/Status,
  $body/Order[1]/Summary[1]/Shipping
  ```

  The above XPaths can be changed to use an intermediate variable:

  ```
  let $summary := $body/Order[1]/Summary[1]
  $summary/Total, $ summary/Status, $summary/Shipping
  ```

  Using intermediate variables consumes more memory but reduces redundant XPath processing.

- Using 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. The hybrid approach includes enabling 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.

  The fields "Total" and "Status" can be retrieved using three `Assign` actions:

  ```
  Assign "$body/Order[1]/Summary[1]" to "foo"
  Assign "$foo/Total" to "total"
  Assign "$foo/Status" to "total"
  ```

# 22

# Oracle Business Intelligence Performance Tuning

This chapter describes tips for tuning Oracle Business Intelligence performance. It contains the following sections:



## 22.1 About Oracle Business Intelligence

Oracle Business Intelligence (BI) Enterprise Edition (or Oracle Business Intelligence) provides a full range of business intelligence capabilities that collects up-to-date data from the organization, presents the data in easy-to-understand formats (such as tables and graphs), and delivers the data quickly to the members of the organization.

These capabilities enable the organization to make better decisions, take informed actions, and implement more-efficient business processes.

## 22.2 Oracle BI Server Query Performance Tuning

This section describes some important considerations for improving query performance with the Oracle BI Server.

For detailed information on BI performance tuning, see "Managing Performance Tuning and Query Caching" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

The following list summarizes methods that you can use to improve query performance:

- **Tuning and indexing underlying databases**: For Oracle BI Server database queries to return quickly, the underlying databases must be configured, tuned, and indexed correctly. Note that different database products have different tuning considerations.

  If there are queries that return slowly from the underlying databases, then you can capture the SQL statements for the queries in the query log and provide them to the database administrator (DBA) for analysis. See "Managing the Query Log" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information about configuring query logging on the system.

- **Aggregate tables**: It is extremely important to use aggregate tables to improve query performance. Aggregate tables contain precalculated summarizations of data. It is much faster to retrieve an answer from an aggregate table than to recompute the answer from thousands of rows of detail.

  The Oracle BI Server uses aggregate tables automatically, if they have been properly specified in the repository. See *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition* for examples of setting up aggregate navigation.

- **Query caching**: The Oracle BI Server can store query results for reuse by subsequent queries. Query caching can dramatically improve the apparent performance of the system for users, particularly for commonly used dashboards, but it does not improve performance for most ad-hoc analysis.

  See "About the Oracle BI Server Query Cache" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information about query caching concepts and setup.

- **Setting parameters in Fusion Middleware Control**: You can set various performance configuration parameters using Fusion Middleware Control to improve system performance. See "Setting Performance Parameters in Fusion Middleware Control" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information.

- **Setting parameters in NQSConfig.INI**: The NQSConfig.INI file contains additional configuration and tuning parameters for the Oracle BI Server, including parameters to configure disk space for temporary storage, set virtual table page sizes, and several other advanced configuration settings. See "NQSConfig.INI File Configuration Settings" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information.

## 22.3 Oracle BI Server Query Cache Performance Tuning

You can configure the Oracle BI Server to maintain a local, disk-based cache of query result sets (query cache). The query cache allows the Oracle BI Server to satisfy many subsequent query requests without having to access back-end data sources (such as Oracle or DB2). This reduction in communication costs can dramatically decrease query response time. See "About the Oracle BI Server Query Cache" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

## 22.4 Oracle BI Web Client Performance Tuning

You can improve the performance of the Oracle BI web client (UI) by configuring your Web server to serve up all static files, as well as enabling compression for both static and dynamic resources. BI 11g ships with WebLogic Server (WLS) serving as the default HTTP server for the BI web client. By allowing the Oracle HTTP Server (OHS) to proxy requests to WLS instead, you may see an improvement in BI Web Client performance. See "Improving Oracle BI Web Client Performance" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

# Part V

# Oracle Identity and Access Management

This part describes tuning the Oracle Identity and Access Management Suite components to improve performance. The Oracle Identity Management products enable you to configure and manage the identities of users, devices, and services across diverse servers. The Access Management products enable you to delegate administration of these identities, and to provide end users with self-service privileges. These products also enable you to configure single sign-on across applications and to process users' credentials to ensure that only users with valid credentials can log into and access online resources.

It contains the following chapters:

# 23

# Oracle Internet Directory Performance Tuning

This chapter provides guidelines for tuning and sizing an Oracle Internet Directory installation. It contains these topics:

- Section 23.1, "About Oracle Internet Directory"
- Section 23.2, "Monitoring Oracle Internet Directory Performance"
- Section 23.3, "Basic Tuning Considerations"
- Section 23.4, "Advanced Tuning Considerations"
- Section 23.5, "Specific Use Cases That Require Additional Tuning"

## 23.1 About Oracle Internet Directory

Oracle Internet Directory is Oracle's Lightweight Directory Application Protocol (LDAP) version 3 Directory Server. Oracle Internet Directory is highly scalable, available, and manageable. It has a multi-threaded, multi-process, multi-instance process architecture with Oracle Database as the directory store. This unique physical architecture enables Oracle Internet Directory to be deployed on several hardware architectures including Symmetric Multi-Processor (SMP), Non-Uniform Memory Access (NUMA) and Cluster hardware. Oracle Internet Directory's physical architecture enables linear performance scalability with hardware resources and numerous high availability configurations.

For more information see *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

---

**Note:** Oracle Internet Directory's out of box configuration is not optimal for most production or test deployments. You must follow at least the steps listed in Section 23.3, "Basic Tuning Considerations" to achieve optimal performance and availability.

---

**See Also:**

- Section 23.2.2, "Obtaining Recommendations by Using the Tuning and Sizing Wizard.".
- "Troubleshooting Oracle Internet Directory" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

Many of the recommendations in this chapter require changes to Oracle Internet Directory system configuration attributes and replication configuration attributes.

**See Also:**

- The "Managing System Configuration Attributes" chapter of *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- The "Managing Replication Configuration Attributes" chapter of *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- The "Attribute Reference" chapter of *Oracle Fusion Middleware Reference for Oracle Identity Management*

for more information about Oracle Internet Directory configuration attributes.

## 23.2 Monitoring Oracle Internet Directory Performance

To identify performance bottlenecks, you can monitor real-time performance metrics for the Oracle Internet Directory database. For more information on how to monitor other Oracle Fusion Middleware components, see Chapter 4, "Monitoring Oracle Fusion Middleware".

### 23.2.1 Monitoring Performance on UNIX and Windows Systems

Knowledge of the following tools is recommended for Linux, Solaris, and other UNIX-like operating systems:

| Tool | Description |
|---|---|
| top | Displays the top CPU consumers on a system |
| vmstat | Shows running statistics on various parts of the system including the Virtual Memory Manager |
| mpstat | Shows an output similar to vmstat but split across various CPUs in the system. This is available on Solaris only. |
| iostat | Shows the disk I/O statistics from various disk controllers |
| sar | Collect, report, or save system activity information. |

Knowledge of the following tools is recommended for Microsoft Windows:

| Tool | Description |
|---|---|
| Windows Performance Monitor | Provides a customized view of the events in the system |
| Windows Task Manager | Provides a high level output (like `top` on UNIX) of the major things happening in the system. |

Knowledge of the following tools is recommended for the Oracle Database:

- `utlbstat.sql` and `utlestat.sql`, or `statspack`
- The ANALYZE function in the DBMS_STATS package

**See Also:**

- *Oracle Database Reference* in the Oracle Database Documentation Library for information about `utlbstat.sql` and `utlestat.sql`
- *Oracle Database Performance Tuning Guide* for information about stats package
- *Oracle Database Concepts* in the Oracle Database Documentation Library for information about the ANALYZE function in the DBMS_STATS package

In addition to the operating system tools, the LDAP applications being used in a customer environment must be able to provide latency and throughput measurement.

In addition, the Database Statistics Collection Tool (oidstats.sql), located at `$ORACLE_HOME/ldap/admin`, is provided to analyze the various database 'ods' schema objects to estimate the statistics. See Section 23.2.3, "Updating Database Statistics by Using oidstats.sql".

## 23.2.2 Obtaining Recommendations by Using the Tuning and Sizing Wizard

Oracle Enterprise Manager Fusion Middleware Control provides a convenient tool for tuning and sizing Oracle Internet Directory.

Use the wizard to obtain tuning and sizing recommendations for your system. You can select Tuning, Sizing, or Both. If you select Sizing or Both, you can select Basic or Advanced

### Tuning

1. From the Oracle Internet Directory menu, select **Administration**, then **Tuning and Sizing**.
2. Click the **Create** icon to invoke the wizard.
3. On the Type Selection page, change the report name, then select **Tuning**.
4. The wizard presents the following pages: Hardware, Features, Load, Data Characteristics, and Garbage Collection.

   On each page, specify values for the text fields (or use defaults) and Select **Yes** or **No** for each question. Some choices might be greyed out, depending upon your previous choices. Most fields have tool tips that appear when you move the cursor over the field.

   Click **Next** to go to the next page or **Back** to return to the previous page. Click **Cancel** to close the wizard.
5. On the Review page, review the data you entered. Click **Back** to change your specifications or click **Finish** to view the report.
6. The report appears on the bottom right section of the page.

   To download the report, click **Download Report**. To delete the report, click **Delete**.

### Sizing

1. From the Oracle Internet Directory menu, change the report name, then select **Administration**, then **Tuning and Sizing**.
2. Click the **Create** icon to invoke the wizard.

3. On the Type Selection page, select **Sizing**.
4. Select **Basic** or **Advanced**.
5. On the Sizing page, specify values for the text fields (or use defaults) and Select Yes or No for each question. Some choices might be greyed out, depending upon your previous choices.
6. Click **Next**.
7. On the Review page, review the data you entered. Click **Back** to change your specifications or click **Finish** to view the report.
8. The report appears on the bottom right section of the page.

   To download the report, click **Download Report**. To delete the report, click **Delete**.

**Both**

1. From the Oracle Internet Directory menu, change the report name, then select **Administration**, then **Tuning and Sizing**.
2. Click the **Create** icon to invoke the wizard.
3. On the Type Selection page, select **Both**.
4. Select **Basic** or **Advanced**.
5. Click **Next**.
6. The wizard presents the following pages: Sizing, Hardware, Features, Load, Data Characteristics, and Garbage Collection.

   On each page, specify values for the text fields (or use defaults) and Select **Yes** or **No** for each question. Some choices might be greyed out, depending upon your previous choices.

   Click **Next** to go to the next page or **Back** to return to the previous page. Click **Cancel** to close the wizard.
7. On the Review page, review the data you entered. Click **Back** to change your specifications or click **Finish** to view the report.
8. The report appears on the bottom right section of the page.

   To download the report, click **Download Report**. To delete the report, click **Delete**.

### 23.2.3 Updating Database Statistics by Using oidstats.sql

Database statistics are updated automatically, OIDMON runs `oidstats.sql` for every configured number of updates to the database. By default, for every 5000 entries added OIDMON runs the `oidstats.sql`. This frequency can be changed using `ldapmodify` commad as shown below

```
$ORACLE_HOME/bin/ldapmodify -p <oidPort> -h <oidHost> -D cn=orcladmin -w
<adminPassword> << eof
dn:  cn=configset,cn=oidmon,cn=subconfigsubentry
changetype: modify
replace: orclstatsperiodicity
orclstatsperiodicity: <desired_number>
eof
```

> **See Also:** The `oidstats.sql` command-line tool reference in *Oracle Fusion Middleware Reference for Oracle Identity Management*

### 23.2.4 Setting Performance-Related Replication Configuration Attributes

To set the replication attributes, you can use either the Replication Wizard in Oracle Enterprise Manager Fusion Middleware Control or the command line.

The attributes `orclthreadspersupplier`, `orclchangeretrycount`, and `orclconflresolution` are replication configuration set attributes.

**See Also:**

- "Configure Replication Attributes by Using Fusion Middleware Control" in*Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- "Configuring Attributes of the Replication Configuration Set by Using ldapmodify" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

for information about

The attributes `orclhiqschedule` and `orclupdateschedule` are replication agreement entry attributes.

**See Also:**

- "Viewing or Modifying an LDAP-Based Replication Setup by Using the Fusion Middleware Control Replication Wizard" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*
- "Configuring Replication Agreement Attributes by Using ldapmodify" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*

**See Also:**

- "Setting Up a One-Way, Two-Way, or Multimaster LDAP-Based Replication Agreement by Using the Replication Wizard in Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* or information on setting replication attributes by using the Replication Wizard.
- "Configuring Attributes of the Replication Configuration Set by Using ldapmodify" in Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory.

### 23.2.5 Managing System Configuration Attributes

You can set most performance-related system configuration attributes from Oracle Enterprise Manager Fusion Middleware Control or from the command line. You can also use the Data Browser in Oracle Directory Services Manager to modify system configuration attributes.

For information on setting system configuration attributes for Oracle Internet Directory, see "Managing System Configuration Attributes" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*:

- "Managing System Configuration Attributes by Using Fusion Middleware Control"

- "Managing System Configuration Attributes by Using WLST"
- "Managing System Configuration Attributes by Using LDAP Tools"
- "Managing System Configuration Attributes by Using ODSM Data Browser"

### 23.2.6 Setting Garbage Collection Configuration Attributes

The attributes `orclpurgetargetage` and `orclpurgeinterval` reside in the changelog purging configuration entry. You can change them with `ldapmodify` or Oracle Directory Services Manager.

#### 23.2.6.1 Modifying Changelog Purging Attributes by Using ldapmodify

The following example is an LDIF file used to configure change log purging.

> **See Also:** "Change Log Purging" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*for a description of change log purging.

This example configures time-based purging for 120 hours (5 days). Use an LDIF file similar to this:

```
dn: cn=changelog purgeconfig,cn=purgeconfig,cn=subconfigsubentry
changetype:modify
replace: orclpurgetargetage
orclpurgetargetage: 240
```

To apply the LDIF file `mod.ldif`, type:

```
ldapmodify -D "cn=orcladmin" -q -p port -h host -D dn -q -f mod.ldif
```

> **See Also:** "Configuring Time-Based Change Log Purging" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

#### 23.2.6.2 Modifying Changelog Purging in Oracle Directory Services Manager

You can modify `orclpurgetargetage` and `orclpurgeinterval` by using the data browser in Oracle Directory Services Manager. You cannot navigate to the changelog purging configuration entry directly in the data tree, but you can get to it by using an advanced search as follows:

1. On the Data Browser tab, click **Advanced**.
2. Expand **Garbage Collection** in the left pane, then select **changelog purgeconfig**. The Garbage Collector Window appears in the right pane.
3. In the right pane, enter the changes you want to make to the **Purge Target Age** and **Purge Interval.**
4. Choose **Apply**.

## 23.3 Basic Tuning Considerations

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

### 23.3.1 Database Parameters

The suggested minimum values for Oracle Database instance parameters are described in Table 23–1:

***Table 23–1 Minimum Values for Oracle Database Instance Parameters***

| Parameter | Value | Notes |
|---|---|---|
| `sga_target` and `sga_max_size` | 1700M for 32-bit systems | Applicable when SGA Auto Tuning using `sga_target` and `sga_max_size` is being used. Especially important for `bulkdelete` performance.<br>A higher value may be required if the directory size exceeds 1 million entries or a high rate of I/O is observed. In case of 64-bit systems, one can go up to 60-70% of the RAM available for the Oracle Database on the box. |
| `db_cache_size` | 1200M for 32-bit systems. | Applicable when SGA Auto Tuning using `sga_target` and `sga_max_size` is not being used. (SGA auto tuning using `sga_target` and `sga_max_size` is recommended instead of this parameter.)<br>A higher value may be required if the directory size exceeds 1 million entries or a high rate of I/O is observed. In case of 64-bit systems, one can go up to 60-70% of the RAM available for the Oracle Database on the box. |
| `shared_pool_size` | 300M | Applicable when SGA Auto Tuning using `sga_target` and `sga_maxsize` is not being used |
| `session_cached_cursors` | 100 | |
| `processes` | 500 | |
| `pga_aggregate_target` | 300M | Before performing a large `bulkload` operation, set this to 1-4GB, if sufficient RAM is available. Set it back after the operation has completed |
| `job_queue_processes` | 1 or more. | Tune this parameter only if you are using Oracle Database Advanced Replication-based multimaster replication |
| `max_commit_propagation_delay` | 99 or lower | Tune this parameter only in Oracle RAC Database deployments, RDBMS v10.1. |

See the *Oracle Database Performance Tuning Guide* for information on setting Oracle Database instance parameters.

### 23.3.2 LDAP Server Attributes

The recommendations in this section are summarized in Table 23–2.

- Tune the number of processes and threads for the Oracle Internet Directory server instance that services LDAP application traffic. This has a major impact on overall performance. See the recommended settings for `orclmaxcc` and `orclserverprocs` in Table 23–2.
- Disable change log generation if you are not deploying either replication or Oracle Directory Integration Platform. Set the attribute `orclgeneratechangelog` to `0`.
- Skip referrals in LDAP searches if you have no referral entries in the directory. Set `orclskiprefinsql` to 1. This can have a major impact on performance.

- Close idle LDAP connections after a period of time instead of leaving them open. This prevents the unnecessary buildup of connections. For example, you can set `orclldapconntimeout` to 60 minutes.

  As of 10*g* (10.1.4.0.1), you can only set this for users who are not configured for operation statistics tracking. Connections by users configured for statistics collection do not time out as per this setting.

  **See Also:** "Configuring a User for Statistics Collection by Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

- If no clients require detailed MatchDN information when the Base DN of an LDAP search operation is not present in the directory, disable it. Change `orclmatchdnenabled` to 0.

The following values are appropriate for most deployments:

***Table 23–2 LDAP Server Attributes to Tune***

| Attribute | Default | Recommended Value | Notes |
|---|---|---|---|
| `orclmaxcc` | 2 | 10 | Server restart required. |
| `orclserverprocs` | 1 | Number of CPU cores on the system | |
| `orclskiprefinsql` | 0 | 1 | This change is highly recommended. Do not change if you have LDAP referral entries. LDAP referral entries are not common.<br>Server restart required. |
| `orclgeneratechangelog` | 1 | 0 | Disable change log generation only if you do not deploy either replication or Oracle Directory Integration Platform. |
| `orclldapconntimeout` | 0 (no timeout) | Varies, 60 minutes is reasonable | Users configured for statistics tracking do not time out. |
| `orclmatchdnenabled` | 1 | 0 | Disable only if no application needs detailed MatchDN information when base DN of a search is not present. |

For information about configuring `orclserverprocs`, `orclldapconntimeout`, and `orclmatchdnenabled` with Oracle Enterprise Manager Fusion Middleware Control, see "Attributes of the Instance-Specific Configuration Entry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

For information about configuring `orclskiprefinsql` or `orclmatchdnenabled` with Oracle Enterprise Manager Fusion Middleware Control, see "Configuring Shared Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

For information about configuring these attributes, as well as `orclgeneratechangelog`, from the command line, see "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

### 23.3.3 Database Statistics

If you use LDAP commands to add a large number entries to Oracle Internet Directory, it can affect directory performance. If this occurs, update the database statistics. See Section 23.2.3, "Updating Database Statistics by Using oidstats.sql."

Typically, you only need to do this when you add entries in bulk for the first time after installing Oracle Internet Directory. You do not need to do it again because the database statistics are updated nightly automatically. If, however, you suddenly experience slow LDAP operations, without a corresponding change in data footprint, consider running `oidstats.sql` once to see if that improves performance. The impact may be due to changes in database SQL execution plans, which oidstats.sql can help to improve.

> **See Also:** *Oracle Database Performance Tuning Guide* for information about SQL tuning.

You do not need to update database statistics if you use the `bulkload` tool to add the entries. The `bulkload` command automatically updates the database statistics.

### 23.3.4 Low-Priority Tuning Considerations

This section describes attributes that can sometimes improve performance, but are considered low-priority.

#### 23.3.4.1 Number of Entries to be Returned by a Search

The attribute `orclsizelimit` controls the maximum number of entries to be returned by a search. The default value is 10000. Setting it very high impacts server performance. It also plays a role in limiting the maximum number of changelogs the replication server can process at a time.

See "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

#### 23.3.4.2 Enabling the Group Cache

The instance-specific subentry attribute `orclenablegroupcache` controls whether privilege groups and ACL groups are cached. Using this cache can improve the performance of access control evaluation for users.

Use the group cache when a privilege group membership does not change frequently. If a privilege group membership does change frequently, then it is best to turn off the group cache. It is important to note that computing a group cache may affect performance. The default is 1 (enabled). Change to 0 (zero) to disable.

See "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

#### 23.3.4.3 Timeout for Write Operations

When an LDAP client initiates an operation, then does not respond to the server for a configured number of seconds, the server closes the connection. The number of seconds is controlled by the `orclnwrwtimeout` attribute of the instance-specific configuration entry. The default is 30 seconds.

You can modify `orclnwrwtimeout` by using Fusion Middleware Control or the command line. See "Attributes of the Instance-Specific Configuration Entry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

## 23.4 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.

- Replication or Oracle Directory Integration Platform
- Replication Server Configuration
- Garbage Collection Configuration
- Oracle Internet Directory with Oracle RAC Database
- Password Policies and Verifier Profiles
- Server Entry Cache
- Result Set Cache
- Tuning Security Event Tracking
- Optimizing Searches

### 23.4.1 Replication or Oracle Directory Integration Platform

When you deploy Oracle Internet Directory with the Oracle Directory Integration Platform or with replication, you can improve performance by having a dedicated LDAP server instance for those two servers. This allows the default Oracle Internet Directory LDAP instance to serve the LDAP application traffic and the second instance to serve LDAP requests from the replication and Oracle Directory Integration Platform servers.

1. Create an additional server instance, as described in the chapter "Managing Oracle Internet Directory Instances" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.
2. Set `orclmaxcc` to 10 and `orclserverprocs` to 1 in the new instance configuration.
3. Restart the server, as described in the chapter "Managing Oracle Internet Directory Instances" in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.
4. Set the SSL and non-SSL ports used by the new instance and configure the replication and Oracle Directory Integration Platform to point to them.

To configure `orclmaxcc` and `orclserverprocs`, see "Attributes of the Instance-Specific Configuration Entry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*. and "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

---

**Note:** In an Oracle Internet Directory Cluster configuration (rack-mounted or multi-box), the replication server must be started on one hardware node only. The LDAP server instance dedicated to replication must be started on the same node. The Oracle Directory Integration Platform server can be on a different node.

---

### 23.4.2 Replication Server Configuration

The following recommendations can be useful when replication traffic is heavy. Be sure you understand the trade-offs before making these changes. The recommended values are summarized in Table 23–3.

- If you are deploying a single master with read-only replica consumers, you may reduce performance impacts by turning off conflict resolution. To do so, change the value of `orclconflresolution` to 0.
- If the supplier is a bottleneck, increase `orclthreadspersupplier` on the supplier. You can also increase `orclthreadspersupplier` at the consumer if is a bottleneck, but be aware that increased parallelism causes race conditions in the application of changelogs, resulting in more human intervention queue (HIQ) changes.
- Decrease `orclchangeretrycount` so that new changelogs get more resources. If there are conflicts, however, this increases the human intervention queue (HIQ) changes.
- Change `orclupdateschedule` to 0 to make the server process changelogs immediately, instead of at the default, 60-second intervals. Do this on both the supplier and consumer.
- Increase the `orclhiqschedule` to a higher value. For example, if accessing the human intervention queue (HIQ) four times a day is sufficient and appropriate for your deployment, set the `orclhiqschedule` to 21600 seconds (6 hours).

Table 23–3 summarizes these recommendations.

***Table 23–3 Replication Attributes***

| Attribute | Default | Recommended Value | Notes |
|---|---|---|---|
| `orclthreadspersupplier` | transport=1<br>apply=5 | Set transport threads to 1 and apply threads to 10 or greater | Most useful if the supplier is the bottleneck. |
| `orclchangeretrycount` | 10 | 4 | Provides more resources to changelogs but might increase HIQ. |
| `orclupdateschedule` | 60 seconds | 0 | Causes changelogs to be processed immediately |
| `orclhiqschedule` | 600 seconds | 21600 seconds | Provides more resources to process new changes. |
| `orclconflresolution` | 1 | 0 | Change only if you are deploying a single master with read-only replica consumers. |

See Section 23.2.4, "Setting Performance-Related Replication Configuration Attributes" for information on setting these replication attributes.

### 23.4.3 Garbage Collection Configuration

By default, Oracle Internet Directory runs database jobs to purge change logs, server manageability statistics, and other data beginning at midnight, with each job starting 15 minutes after the previous one. You can change this configuration to suite your deployment needs by modifying the parameters shown in Table 23–4.

*Table 23–4 Garbage Collection Configuration Parameters*

| Parameter | Value | Notes |
|---|---|---|
| `orclpurgetargetage` | Less than 10days (240 hours) | Only if there is no requirement to retain change logs |
| `orclpurgeinterval` | 6–12 hours | |

You can modify these attributes by using `ldapmodify` or Oracle Directory Services Manager. See Section 23.2.6, "Setting Garbage Collection Configuration Attributes."

### 23.4.4 Oracle Internet Directory with Oracle RAC Database

As described in Section 23.4.2, "Replication Server Configuration", you can have a dedicated LDAP server for Oracle Directory Integration Platform and replication, in addition to the default server. In an Oracle Internet Directory Cluster, start the default LDAP instance on all Oracle Internet Directory nodes, but start the dedicated instance only on the node where Oracle Directory Integration Platform and replication are running.

Consider carefully which database instance Oracle Internet Directory should connect to:

- You can configure the Oracle Internet Directory for load balancing between Oracle Database instances in the cluster, or failover mode.
- If you use a dedicated LDAP server instance for replication and Oracle Directory Integration Platform, you can configure the connection strings of that instance for failover. You would use the following in `tnsnames.ora`:

  ```
  (FAILOVER=ON)(LOAD_BALANCE=OFF)
  ```

- When performing a bulk operation, such as `bulkload`, connect the tool to just one Oracle Database instance for the entire operation.
- Configure Oracle Internet Directory instances as follows:
  - One Oracle Internet Directory instance on each of the nodes to service LDAP application traffic
  - An instance of the Oracle Internet Directory replication server and Oracle Directory Integration Platform server on one node

### 23.4.5 Password Policies and Verifier Profiles

Oracle Internet Directory has password policies and password verifier profiles enabled out of box. If Oracle Internet Directory is not required to enforce password policies in a given deployment, then the password policies can be disabled. The password verifier profiles enabled out of box control the generation of certain password verifiers required by Oracle products like Enterprise User Security and Oracle Collaboration Suite. If Oracle Internet Directory is not being deployed for other Oracle products, you can disable all the password verifier profiles.

You can disable password policies and password verifiers by using Oracle Directory Services Manager or `ldapmodify`.

**See Also:**

- The "Managing Password Policies" chapter in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.
- The "Managing Password Verifiers" chapter in *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

## 23.4.6 Server Entry Cache

The Oracle Internet Directory server entry cache enables LDAP entries to be cached on the Oracle Internet Directory server process heap for better performance. Configuring the entry cache provides benefits if, and only if, all or most entries can be cached.

**Caution:** The server entry cache is beneficial for small directory deployments only. Some of the tuning recommendations here contradict the tuning recommendations in the earlier sections. Review the applicability of entry cache to a given deployment and incorporate the tuning mentioned in this section only if all considerations enumerated here are met.

### 23.4.6.1 Benefits of Using the Entry Cache

One of the key benefits of using the entry cache is that the LDAP search operations with base scope are about five times as fast. This applies only when all or most entries can be cached. A cache miss is more expensive than disabling the entry cache.

### 23.4.6.2 Values for Configuring the Entry Cache

You can configure and optimize the server entry cache by setting the values shown in Table 23–5.

***Table 23–5 Server Entry Cache Configuration***

| Attribute | Default | Recommended Value | Notes |
|---|---|---|---|
| `orclmaxcc` | 2 | 10 | Restart the server after changing this attribute. |
| `orclserverprocs` | 1 | Total number of cores on the system. | |
| `orclecacheenabled` | 1 | 1 | |
| `orclecachemaxsize` | 200000000 Bytes | Total size of the directory, in bytes | To determine the optimal setting for this attribute, use the number of entries in the Directory Information Tree and multiply by the average entry size.<br>Estimate three times the size of the entries in LDIF format. |
| `orclecachemaxentries` | 100000 | Total number of entries in the DIT | |
| `orclecachemaxentsize` | 1000000 | Size, in bytes, of the largest entry in the DIT | The largest entry is usually a group entry or an entry with binary attribute values. |

For example, if the total size of the Directory Information Tree is 300K and the total size of 300K entries in LDAP Data Interchange Files (LDIF) format is 500M, you would set `orclecacheenabled` to 1, `orclecachemaxsize` to 1,500,000,000, and `orclecachemaxentries` to 300,000. If the size of the largest group entry or entry with binary value is 10M, you would set `orclecachemaxentsize` to 10,000,000.

To obtain the number of entries in the Directory Information Tree, use the following command:

```
sqlplus ods@oiddb
select count(*) from ct_dn;
```

```
oidctl connect=oiddb status -diag
```

The following example shows the `oidctl connect=oiddb status -diag` command output:

```
+-------------------------------------------------------------------------+
| Process      |  PID   |   InstName    |  CompName   |Inst#| Port | Sport |
+-------------------------------------------------------------------------+
| oidmon       |   8192 |         inst1 |        oid1 |    0|      |       |
+-------------------------------------------------------------------------+
| oidldapd disp|   8201 |         inst1 |        oid1 |    1| 5678 |     0 |
| oidldapd serv|   8205 |         inst1 |        oid1 |    1| 5678 |     0 |
| oidldapd serv|   8209 |         inst1 |        oid1 |    1| 5678 |     0 |
| oidldapd serv|   8213 |         inst1 |        oid1 |    1| 5678 |     0 |
| oidldapd serv|   8217 |         inst1 |        oid1 |    1| 5678 |     0 |
| Config   DN  | cn=oid1,cn=osdldapd,cn=subconfigsubentry                  |
+-------------------------------------------------------------------------+


+-------------------------------------------------------------------------+
|Printing LDAP Operation in progress status ...                           |
+-------------------------------------------------------------------------+
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8205 WorkerID: 8 DBSID: 168 DBPID: 8245 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8205 WorkerID: 9 DBSID: 170 DBPID: 8253 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8205 WorkerID: 10 DBSID: 180 DBPID: 8261 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8205 WorkerID: 11 DBSID: 189 DBPID: 8269 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8209 WorkerID: 13 DBSID: 171 DBPID: 8249 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8209 WorkerID: 9 DBSID: 181 DBPID: 8257 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8209 WorkerID: 12 DBSID: 193 DBPID: 8267 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8209 WorkerID: 10 DBSID: 199 DBPID: 8225 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8209 WorkerID: 11 DBSID: 190 DBPID: 8227 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8205 WorkerID: 13 DBSID: 197 DBPID: 8223 ==> IDLE
+-------------------------------------------------------------------------+
  OIDLDAPD_PID: 8205 WorkerID: 12 DBSID: 182 DBPID: 8229 ==> IDLE
+-------------------------------------------------------------------------+

        Cache Max Size                     : 1000000512
        Max Entries configured            : 1000000
        Max Entries cached                 : 100000
        Num Entries in Cache              : 100000
```

```
Num Entries in GC                  : 0
Page size                                 : 976556
Entry cache Hit count               : 6172127
Entry cache Mis count              : 99999
Hash Area bytes used              : 24497696
Hash Area blocks used            : 37
ResultSet cache bytes used      : 6799604
Resultset cache blocks used     : 300000
Entry cache bytes used            : 404047820
Entry cache blocks used          : 5900293
Cache memory used                : 435345120
```

To configure the attributes, see "Attributes of the Instance-Specific Configuration Entry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* and "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

## 23.4.7 Result Set Cache

Result set cache is an Oracle 11g OID feature that allows complete result sets to be stored in memory. If a SQL query is executed and its result set is in the cache then almost the entire overhead of the SQL execution is avoided: this includes parse time, logical reads, physical reads and any cache contention overhead (latches for instance) that might normally be incurred. Configuring the result cache can improve performance since most LDAP applications typically look up user entry such as *mail=john.doe@acme.com* or *uid=john.doe* from a user tree. Such queries are repeated by the application every time a user logins or uses the application. The result set of such queries may be a single entry. Performance may be affected as OID makes a trip to the database for the entry each time the query is run.

### 23.4.7.1 When to Use Result Set Cache

Consider using Result Set Cache only under the following conditions:

- Filter matches one or few entries.
- SQL statement causes multiple reads from disk or buffer (expensive)

### 23.4.7.2 Benefits of Using Result Set Cache

Benefits of using the entry cache include:

- OID evaluates the filter without making a trip to the database and therefore reduces the load on the database.

  Note that the result set cache database parameter can be configured on the client side or server side. When the server side cache is enabled, the result set cache can consume a significant amount of database memory and OID performance may be impacted.

- Performance improved by 3 to 5 times when compared to performance when result set cache is not used.

### 23.4.7.3 Values for Configuring Result Set Cache

Note that any change to the following configuration attributes requires a restart of OID server (all the instances).

*Table 23–6 Result Set Cache Attributes to Tune*

| Attribute | Default | Recommended Value | Notes |
|---|---|---|---|
| `OrclRSCacheAttr` | cn, mail, uid, orclguid | | Multi valued attribute, Value contains the name of the Attribute. Typically these attributes are not modified for the life of the entry. |
| `ResultSetMaxEntries` | 4 | | Maximum number of entries for a given search that can be cached. |
| `ResultSetMaxCacheSize` | 10 MB | | Maximum memory that can be allocated in the shared memory for the result set cache. |
| `ResultSetMaxTime` | 8 hours | | Time to live for the result set cache when the cache is full. |

## 23.4.8 Tuning Security Event Tracking

The instance-specific configuration entry attributes `orcloptrackmaxtotalsize` and `orcloptracknumelemcontainers` control how much memory is used for security event tracking.

The attribute `orcloptrackmaxtotalsize` specifies the maximum number of bytes of RAM that security events tracking can use for each type of operation. If the Directory Server exceeds this limit for information collected for an operation, the server stops collecting new information and records appropriate messages in server log files. For the compare operation, the Directory Server uses twice the value of the attribute, which is the combined amount of information about users performing compare operation and users whose passwords are being compared. The default value of `orcloptrackmaxtotalsize` is 100000000 Bytes, which should be sufficient for most deployments. It can be increased to 200MB. For information about modifying `orcloptrackmaxtotalsize`, see the instance-specific configuration attribute examples in "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

The attribute `orcloptracknumelemcontainers` allows you to choose the number of in-memory cache containers to be allocated for security event tracking in the Oracle Internet Directory server. There are two subtypes for this attribute. They are `1stlevel` and `2ndlevel`. The `1stlevel` subtype is for setting the number of in-memory cache containers for storing information about users performing operations. The `2ndlevel` subtype, which is applicable only to compare operation, sets the number of in-memory cache containers for information about the users whose user password is compared and tracked when detailed compare operation statistics is programmed.

The default value of both subtypes is `256`. The appropriate values for these subtypes depend on the number of users in your environment and the number of applications used to access the directory, as follows:

- In a deployment where several applications perform operations on behalf of a large number of end users, set 1stlevel proportional to the number of applications, plus a few hundred more for end users directly accessing the directory. Then set 2ndlevel proportional to the number of end users.
- In a deployment where end users themselves perform the operations, set `1stlevel` proportional to the number of end users, then set `2ndlevel` to a small value, such as 25.

- A typical proportional value is one fifth. Proportions between one tenth and one half are reasonable in most environments.

If your deployment requires it, set the values for `orcloptracknumelemcontainers` only when security events collection is turned on.

## 23.4.9 Optimizing Searches

This section contains these topics:



### 23.4.9.1 Optimizing Searches for Large Group Entries

Searches for group entries with several thousand attribute values for either the `member` or `uniquemember` attribute can have high latency. If you find the latency unacceptably high, there are steps you can take to reduce it.

The simplest step is to reduce the number of attributes you are searching for. If you do not need to retrieve all the attributes of the group entry, specify required attributes in the search request to optimize the latency.

**23.4.9.1.1 Entry Cache Enabled Configuration** If you still see unacceptable latency, even with required attributes specified, then you can try to cache the large group entry in the entry cache. To do this, increase the value of the `orclEcacheMaxEntSize` attribute in the instance-specific configuration entry:

`cn=`*`componentname`*`,cn=osdldapd,cn=subconfigsubentry`

This attribute controls the maximum size of a cache entry.

> **Note:** If you expect frequent updates to large groups, then do not use this tuning methodology. Use the Entry Cache Disabled Configuration.

**23.4.9.1.2 Entry Cache Disabled Configuration.** No action is required. This configuration is enabled by default.

### 23.4.9.2 Optimizing Searches for Skewed Attributes

To service a typical search request, the Directory Server sends a SQL statement to the Oracle Database. If a given attribute has very different response times depending on its value, then the attribute is said to be skewed. For example, if searches for `my_attribute=value1` and `my_attribute=value2` have very different response times, then `my_attribute` is said to be a skewed.

You can uniform the response times for searches for such an attribute by adding it as a value of the `orclskewedattribute` attribute, which is in the DSA configuration entry. The DN of the DSA configuration entry is

`cn=dsaconfig,cn=configsets,cn=oracle internet directory`

By default, the `objectclass` attribute is listed as a value in the `orclskewedattribute` attribute.

You can change the value of `orclskewedattribute` by using or `ldapmodify`. See "Attributes of the Instance-Specific Configuration Entry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* and "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

#### 23.4.9.3 Optimizing Performance of Complex Search Filters

When Oracle Internet Directory receives an LDAP search filter from a client application, it sends the filter to the Oracle Database as an SQL query. Sometimes client applications send filters that include terms that match a large number of entries in the directory. For example, consider the following filter:

```
(&(uid=msmith)(objectclass=inetorgperson)(orclisenabled=TRUE))
```

The terms `(objectclass=inetorgperson)` and `(orclisenabled=TRUE)` in that filter match nearly all entries. It would be very resource-intensive to execute that entire filter in the Oracle Database. To improve performance, you can specify that Oracle Internet Directory execute a portion of that filter in its own memory, rather than in the database. To do that, you use `orclinmemfiltprocess`, an attribute in the DSA configuration entry:

```
cn=dsaconfig,cn=configsets,cn=oracle internet directory
```

When `orclinmemfiltprocess` is configured, the following events occur each time Oracle Internet Directory receives an LDAP search:

1. Oracle Internet Directory removes all the terms that are configured in the `orclinmemfiltprocess` before forming the SQL query.
2. Oracle Internet Directory sends the SQL query to Oracle Database.
3. Oracle Database sends the entries resulting from the SQL query to Oracle Internet Directory.
4. Oracle Internet Directory applies the original filter sent by the client (the terms in `orclinmemfiltprocess`) to those entries in memory.
5. Oracle Internet Directory sends the entries that match that filter to the client.

For example, suppose `orclinmemfiltprocess` is set to `(objectclass=inetorgperson)(orclisenabled=TRUE)`. When Oracle Internet Directory receives the search `(&(uid=msmith)(objectclass=inetorgperson)(orclisenabled=TRUE))`, it sends a filter containing only the parameter `(uid=msmith)` to the database. After Oracle Internet Directory receives entries back from the database, Oracle Internet Directory itself applies the filter `(objectclass=inetorgperson)(orclisenabled=TRUE)` to those entries.

By default, `orclinmemfiltprocess` is set to the following values:

```
(objectclass=inetorgperson)

(objectclass=oblixorgperson)

(|(!(obuseraccountcontrol=*))(obuseraccountcontrol=activated))

(|(obuseraccountcontrol=activated)(!(obuseraccountcontrol=*)))

(objectclass=*)

(objectclass=oblixworkflowstepinstance)

(objectclass=oblixworkflowinstance)
```

```
(objectclass=orcljaznpermission)

(obapp=groupservcenter)(!(obdynamicparticipantsset=*))

(objectclass=orclfeduserinfo)
```

You can change the value of `orclinmemfiltprocess` by using or `ldapmodify`. See "Attributes of the Instance-Specific Configuration Entry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* and "Setting System Configuration Attributes by Using ldapmodify" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

Under some conditions, Oracle Internet Directory ignores `orclinmemfiltprocess` and sends the entire filter to the database. It does this if the filter it receives meets the following conditions:

- It contains only one parameter, that is, one attribute-value pair.
- It contains no filter condition other than those in `orclinmemfiltprocess`
- It contains an OR condition applied to the terms that are in `orclinmemfiltprocess`
- It contains the same terms as in `orclinmemfiltprocess`, but in a different order

The following cases illustrate those conditions. In all of the following cases, `orclinmemfiltprocess` is set to `(objectclass=inetorgperson)(employeetype=Contract)`.

### Examples

#### Case A

```
(&(manager=cn=john doe)(objectclass=inetorgperson)
(employeetype=Contract))
```

Oracle Internet Directory sends the filter `(&(manager=cn=john doe))` to the database.

#### Case B

```
(&(uid=rmsmith)((objectclass=inetorgperson)(employeetype=Contrac
t)))
```

Oracle Internet Directory sends only `(&(uid=rmsmith))` to the database, then applies the filter `(&(objectclass=inetorgperson)(employeetype=Contract))` to the entries that are returned from the database.

#### Case C

```
(|(uid=rmsmith)(objectclass=inetorgperson)
(employeetype=Contract))
```

In this filter, the terms that match `orclinmemfiltprocess` are part of an OR condition. Oracle Internet Directory sends the filter, as is, to the database.

#### Case D

```
(&(uid=rmsmith)(employeetype=Contract)
(objectclass=inetorgperson))
```

Even though some of the terms in this filter match `orclinmemfiltprocess`, they are in a different order, so Oracle Internet Directory sends the whole filter to the database. You could add `(employeetype=Contract)(objectclass=inetorgperson)` to `orclinmemfiltprocess` if you do not want Oracle Internet Directory to send this filter to the database.

#### Case E

```
(|(&(uid=rmsmith)(sn=smith)(objectclass=inetorgperson)(employeet
ype=Contract))
```

In this filter, the terms that match `orclinmemfiltprocess` are part of an OR condition. Oracle Internet Directory sends the filter, as is, to the database.

#### Case F

```
(&(|(uid=rmsmith)(sn=smith))(objectclass=inetorgperson)(employee
type=Contract)))
```

Even though this filter contains an OR operator, it is not applied to the terms that match `orclinmemfiltprocess`. Oracle Internet Directory sends `(&(|(uid=rmsmith)(sn=smith)))` to the directory and applies the filter `(&(manager=cn=john doe)(&(objectclass=inetorgperson)(employeetype=Contract))` to the entries that are returned from the database.

#### Configuring Multiple Filters

If the application is sending multiple filters, and the terms in one filter are a superset of the terms in the other, you must configure `orclinmemfiltprocess` for both values.

For example, suppose the application is sending the following two filters:

```
(&(uid=rmsmith)(objectclass=inetorgperson)(employeetype=Contract
))
```

```
(&(uid=rmsmith)(objectclass=inetorgperson)(employeetype=Contract
)(departmentNumber=627))
```

where `(departmentNumber=627)` matches a lot of entries. You must configure `orclinmemfiltprocess` as follows:

```
(objectclass=inetorgperson)(employeetype=Contract)
```

```
(departmentNumber=627)
```

#### Optimizing Performance for Search baseDN

In the DIT, if all the users are under one baseDN, such as `cn=users,dc=acme,dc=com`, and all the LDAP search clients send base as `cn=users,dc=acme,dc=com`, then the configuration of the `orclinmemfilter` will significantly reduce database processing time. See the following example:

```
orclinmemfiltprocess;dn: cn=users,dc=acme,dc=com
```

## 23.5 Specific Use Cases That Require Additional Tuning

This section describes some specific use cases that require additional tuning, in addition to Section 23.3, "Basic Tuning Considerations".

### 23.5.1 Bulk Load Operations

If you are planning a large `bulkload` operation, make the following changes:

- Set the database initialization parameter `pga_aggregate_target` to 1-4GB for the duration of the operation, if sufficient RAM is available.
- Increase the database temporary tablespace before loading a large number entries. You need about 1G of temporary tablespace per million entries being loaded. You can free up the tablespace after the operation.

### 23.5.2 Bulk Delete Operations

If you are planning a large `bulkdelete` operation, perform the following tasks:

- Ensure that the database initialization parameter `sga_target` are tuned as described in Section 23.3.1, "Database Parameters."
- Set the database initialization parameter `log_buffer` to 10M. This can provide additional performance benefit.
- Ensure that you have at least three database redo log files with at least 100MB.
- Ensure that the undo tablespace is at least 1 GB in total size.
- Follow the recommendations about redo logs and undo tablespace in the next section, Section 23.5.3, "High LDAP Write Operations Load."

### 23.5.3 High LDAP Write Operations Load

If you have a high LDAP write operations load, or if you perform many `bulkdelete` operations, consider tuning the following values:

- Increase the size or number of the database redo log files so that the total size is 1000-1500 MB. Other considerations affect the total size of redo logs.
- Depending on how the disks are configured, it might be beneficial to isolate the redo log files to a dedicated set of disks.
- Increase the undo tablespace size by adding data files to this tablespace. For most deployments, 2-4 GB should suffice.
- Do not use the Oracle Internet Directory server entry cache. See Section 23.4.6, "Server Entry Cache."
- If neither Oracle Internet Directory replication nor DIP is deployed, disable change log generation. See Section 23.4.1, "Replication or Oracle Directory Integration Platform."

Table 23–7 summarizes the redo log and undo tablespace recommendations provided in this section.

***Table 23–7 Redo Log and Undo Tablespace Values***

| Attribute | Value | Notes |
|---|---|---|
| Redo Log | 3 logs, 100MB each | Many `bulkdelete` operations. |
| Redo Log | Total size 1000-15000MB | Large number of write operations. |
| Undo Tablespace | At least 1GB total | Many `bulkdelete` operations. |

*Table 23–7 (Cont.) Redo Log and Undo Tablespace Values*

| Attribute | Value | Notes |
|---|---|---|
| Undo Tablespace | 2-4 GB | Large number of write operations. |

# 24

# Oracle Virtual Directory Performance Tuning

This chapter provides tuning tips for Oracle Virtual Directory. It contains the following sections:



## 24.1 About Oracle Virtual Directory

Oracle Virtual Directory is an LDAP Version 3-enabled service that provides an abstracted view of one or more enterprise data sources. Oracle Virtual Directory consolidates multiple data sources into a single directory view, enabling you to integrate LDAP-aware applications with diverse directory server data stores.

The information in this chapter assumes that you have reviewed the concepts and administration information in the *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

---

**Note:** Oracle Virtual Directory's out of box configuration may not be optimal for many production and test deployments. You are encouraged to incorporate the recommendations listed in "Basic Tuning Considerations" to achieve optimal performance and availability.

---

## 24.2 Basic Tuning Considerations

The tuning considerations in this section apply to most deployments and usage scenarios. It is highly recommended that you review these configurations and implement those that are appropriate for your use case scenarios. The tuning information is summarized in Table 24–1.

*Table 24–1 Basic Tuning Considerations*

| Configuration Attribute | Category | Default Value | Recommended Value | Notes |
|---|---|---|---|---|
| Threads | Listener Properties | 10 | 10 * Number Of central processing units (CPUs) available for Oracle Virtual Directory Server | Recommendation applies only to the active LDAP Listeners.<br>For more information, see Tuning Worker Threads. |
| Work Queue Capacity | Listener Properties | 2048 | Expected Number of Max Concurrent Clients * 2 | 2048 operations are executed concurrently. Some clients may send asynchronous operations as well.<br>For more information, see Tuning Work Queue Capacity. |
| Max, Initial Pool Connections | LDAP Adapter Properties | 10 | Total Number of 'Threads parameter values for all active Listeners that use this Adapter | Ensure that the back-end Directory Servers can handle these connections.<br>For more information, see Tuning the LDAP Connection Pool. |
| Max Heap Size | System Properties | 256 MB | Up to 2 GB on 32-bit systems and higher values on 64-bit systems. | Higher values protect against Out Of Memory errors. Ensure that there is sufficient RAM on the system to handle the configured value.<br>For more information, see Tuning Heap Size. |

## 24.2.1 Tuning the Ping Interval

Consider increasing the ping interval to 60 seconds (or more as needed) in the opmn.xml file.

When the system is busy, a ping from the Oracle Process Manager and Notification Server (OPMN) to Oracle Virtual Directory may fail. As a result, OPMN will restart Oracle Virtual Directory after 20 seconds (the default ping interval). To avoid this, consider increasing the ping interval to 60 seconds or more.

The ping interval can be modified in the `$ORACLE_INSTANCE/config/OPMN/opmn/opmn.xml` as shown below:

```
<process-type id="OVD" module-id="OVD">
    <module-data>
        <category id="start-options">
                <data id="java-bin" value="$ORACLE_HOME/jdk/bin/java"/>
         <data id="java-options" value="-server -Xms2056m -Xmx2056m
```

```
-Dvde.soTimeoutBackend=0 -DdisableECID=1 -Didm.oracle.home=$ORACLE_HOME
-Dcommon.components.home=$ORACLE_HOME/../oracle_common
-Doracle.security.jps.config=$ORACLE_INSTANCE/config/JPS/jps-config-jse.xml"/>
          <data id="java-classpath" value="$ORACLE_HOME/ovd/jlib/vde.jar$:$ORACLE_
HOME/jdbc/lib/ojdbc6.jar"/>
      </category>
   </module-data>
     <stop timeout="120"/>
     <ping interval="60"/>
</process-type>
```

### 24.2.2 Tuning Worker Threads

Tune the number of worker threads based on the number of central processing units (CPU) available for Oracle Virtual Directory Server on the system.

The 'Threads' configuration parameter in the Oracle Virtual Directory Listener settings should be set to an appropriate value. The default value for Threads in the Admin Gateway listener and DSML Gateway listener should not be changed. The number of Threads for the LDAP Listeners are typically the threads that need to be tuned since it is the LDAP Listeners that take on concurrent traffic from applications. A common configuration is to have 10 threads per CPU. For example, if there are 4 central processing units on the system, then there would be 40 threads.

For more information, see "Managing Listeners" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

### 24.2.3 Tuning Work Queue Capacity

Tune the Work Queue Capacity based on the expected maximum number of concurrent clients to a given LDAP Listener.

The 'WorkQueueCapacity' configuration parameter in the Oracle Virtual Directory Listener settings should be set to an appropriate value. This ensures that the connection requests from LDAP clients are not rejected due to a lack of work queue capacity. Work elements are allocated on demand only, therefore a value higher than the actual estimate can be used.

The Fusion Middleware Control Performance Monitor provides a historical report which contains the maximum number of connections. Use this report to determine how to adjust the connection value based on production data.

If Oracle Virtual Directory needs to support high number of concurrent clients, then set the ulimit 'nofiles' (descriptor) parameter to the number of LDAP Clients expected. For example, in the command window where OPMN is started, set the following `ulimit` when 8000 concurrent clients are expected:

```
ulimit -n 8192
```

This change requires restart of OPMN and Oracle Virtual Directory to take effect.

For more information, see "Managing Listeners" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

### 24.2.4 Tuning the LDAP Connection Pool

Tune the size of the LDAP connection pool in Oracle Virtual Directory LDAP Adapter to be at least as high as the total number of Threads configured in the Oracle Virtual Directory Listeners that actively use the LDAP Adapter.

This ensures that in the worker threads have enough LDAP connections to process requests. The actual number of active adapters, active listeners and traffic pattern control the usage of connections. However, since connections that are idle in the LDAP Adapter connection pool are periodically closed, a higher value should not impact performance. Ensure that the back-end Directory Server is configured to handle the number of concurrent connections from Oracle Virtual Directory LDAP Adapter connection pool.

For more information, see "Configuring LDAP Adapter" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

### 24.2.5 Tuning Heap Size

Tune the maximum Java heap size of the JVM running Oracle Virtual Directory. This is to ensure that Oracle Virtual Directory has sufficient heap to handle the concurrent load.

For more information, see "Controlling the Maximum Heap Size Allocated to the Oracle Virtual Directory Server" in *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory*.

## 24.3 Advanced Tuning Considerations

Depending on your Oracle Virtual Directory deployment's use case scenarios, the following tuning configurations may improve performance.

### 24.3.1 Tuning Database Adapters

The Database Adapter is a fully featured LDAP-to-JDBC gateway supporting translation of all LDAP operations (add, bind, delete, baseSearch, modify, wildCardSearch) into equivalent SQL prepared statement code. The Database Adapter uses JDBC class libraries to form connections to databases for the purpose of performing LDAP searches. The database libraries are generally provided by the database vendor.

> **Note:** For improved performance, tune the database before using the Database adapter. Consult your database documentation for more information. If the database being used is an Oracle database, see *Oracle Database Performance Tuning Guide*.

For optimal performance, consider the following configuration options for the database schema against which the Oracle Virtual Directory database adapter is configured:

- In general, the mapped columns in the underlying database schema should have an index defined if the mapped LDAP attribute is used in LDAP search filters.
- In scenarios where an LDAP attribute that is used in an LDAP search filter has a matching rule of 'caseIgnoreMatch', the mapped database table column for this attribute needs a function index to be defined for optimal look-up performance.

  For example, if LDAP attribute 'CN' is mapped to database schema column EMP.NAME, then a function index on UPPER(EMP.NAME) is required for optimal performance of LDAP search filters involving CN attribute.

  For more information on function-based indexes, see "Using Function-based Indexes for Performance" in *Oracle Database Performance Tuning Guide*.

Table 24–2 describes some additional Database Adapter settings:

***Table 24–2 Database Adapter Settings***

| Parameter | Value | Notes |
|---|---|---|
| Adapter | Default: Active | An adapter can be configured as Active or Inactive. An inactive adapter can not start during a server restart or when you try to start it. The purpose of the Inactive setting is to keep old configurations available or on stand-by without having to delete them from the configuration. |
| Maximum Connections | Default: 10 connections | This defines the maximum connections the Database Adapter may make with the database. |
| Database Connection Timeout | Default: 10 seconds | The database connection timeout adapter property controls the LDAP request to wait for a connection to become available in the cache after reaching the maximum number of connections limit.<br>If a connection does not become available within the number of seconds defined, the LDAP request fails. If database connection timeout system property is not used, the LDAP request waits 10 seconds for a connection to become available. |

## 24.3.2 Tuning Join Adapters

If you are using Join Adapters, join only appropriate sources. For example if a deployment requires only to link attributes in the primary source under "cn=users" branch, create a primary adapter that only exposes this branch. And then create the join rule with that adapter. This can reduce the need for Oracle Virtual Directory to try to join entries that may never have corresponding linked entries.

> **Tip:** Always make sure that the attributes used by join rules are properly indexed.

## 24.3.3 Tuning Filters

If a known client search filter does not apply to certain adapters, apply the filter to all applicable "Exclude Filters" to improve performance and reduce network traffic.

## 24.3.4 Tuning Load Balancer Local Store Adapter

Some load balancers query an LDAP server to determine if it is up or down. If your load balancer uses this feature, consider creating a local store adapter with a separate namespace (for example dc=loadbalancer) that is used only for the load balancer. While the performance impact of the load-balancer is probably not noticeable, by keeping it in a separate namespace. it makes it easier to exclude the load-balancer `KeepAlive` requests from creating large log files during troubleshooting.

## 24.3.5 Tuning the Cache Plug-In

The CachePlug-in provides an in-memory cache for Oracle Virtual Directory. It has the ability to cache query results from any source for re-use by LDAP clients. This plug-in can improve performance for those applications where queries are highly repetitive.

To review cache operation and configuration, set VE logging level to 'Dump' to see more details. Because the cache is a normal plug-in, the cache can be configured to run anywhere within Oracle Virtual Directory. It can be executed globally, or within the context of a single adapter. It can also be restricted to specific namespaces by using the namespace filtering available in standard plug-in configuration.

#### 24.3.5.1 Cache Hit Logic

The cache works by storing query results and making them available for later use. If a query is repeated by the same user and the same attributes or a subset of attributes are requested, the cache can return its results instead of having Oracle Virtual Directory pull the information from the source. The plug-in can also be configured to allow cache hits to be shared between users.

Sharing cache entries between users should not be used unless the pass credentials are not being passed to back-end sources and Oracle Virtual Directory is solely responsible for security enforcement. Careful consideration should be given when sharing cache hits between users as it would then be possible for one user to see something they should not, since they may have access to a cache result from a more privileged user.

#### 24.3.5.2 Cache Plug-in Memory Management

This plug-in periodically reviews the cache and checks for expired results, or entries that have been invalidated by a previous modify transaction. In the event that the cache quota is exceeded, the plug-in attempts to trim memory by purging the queries that were least recently used (LRU).

Table 24–3 describes some parameters used to tune the Memory Management Plug-in:

***Table 24–3 Memory Management Plug-in Settings***

| Parameter | Value | Notes |
|---|---|---|
| Size | Default: 1000 entries | The maximum number of entries that may be cached at any one time. |
| MaxResultSize | Default: 1000 entries | The maximum number of entries that may be cached for any particular query. |
| Trimsize | Default: 1000 entries | When the maximum cache size is exceeded, the amount by which the cache manager must reduce the balance. Note: when necessary, trimming is done by purging expired queries first followed by queries in order of least recent use. |
| MaximumAge | Default: 600 seconds | The maximum age in seconds for any query/entry stored in the cache. |
| MaintenanceInterval | Default: 60 seconds | The interval in seconds between when the cache manager checks for expired queries. |
| BySubject | Default: 1 (not shared) | A flag (1 or 0) indicating whether cache results are shared between subjects. A value of 1 indicates that results are not be shared between subjects. |

### 24.3.6 Tuning LDAP Listener

Table 24–4 describes some parameters used to tune the LDAP Listener:

***Table 24–4 Listener Parameters***

| Parameter | Value | Notes |
|---|---|---|
| Backlog | Default: 128 requests | Specifies the maximum number of pending connection requests that are allowed to queue up before the server starts rejecting new connection attempts.<br>The default value is sufficient in most cases and the need to change this value is very rare. |
| Reuse address | Default: False | This option determines whether LDAP listener should reuse socket descriptors.<br>If enabled, the SO_REUSEADDR socket option is used on the Oracle Virtual Directory server listen socket to potentially allow the reuse of socket descriptors for clients in TIME_WAIT state. |
| Keep Alive | Default: False | This option determines whether the LDAP connection should use TCP keep-alive.<br>If enabled, the SO_KEEPALIVE socket option is used to indicate that TCP keepalive messages should periodically be sent to the client to verify that the associated connection is still valid. |
| TCP No delay | Default: True | This option determines whether the LDAP connection should use TCP no-delay.<br>If enabled, TCP_NODELAY socket option is used to ensure that response messages to the client are sent immediately rather than potentially waiting to determine whether additional response messages can be sent in the same packet. |

*Table 24–4 (Cont.) Listener Parameters*

| Parameter | Value | Notes |
|---|---|---|
| Read Timeout | Default: 0 | This option enables/disables SO_ TIMEOUT with the specified timeout, in milliseconds.<br>With this option set to a nonzero timeout, client connection to the Oracle Virtual Directory server can remain idle only for this amount of time. If the connection is idle for a period longer than the specified timeout, the client connection is terminated.<br>A timeout of zero is interpreted as an infinite timeout.<br>**Warning:** This option is equivalent to vde.soTimeoutFrontend system property in Oracle Virtual Directory version 10g. The vde.soTimeoutFrontend system property is not supported for 11g. Users must modify the value specified in system property<br>The mapping of values from 10g to 11g are:<br>.Enabled to 0<br>Disabled to nonzero amount of time in milliseconds |

## 24.3.7 Tuning the Server for OVD

Table 24–5 describes some basic parameters used to tune the server:

*Table 24–5 Server Parameters*

| Parameter | Value | Notes |
|---|---|---|
| Anonymous Search Limit | Default: 1000 | The maximum number of entries returned for an anonymous client. |
| Connection Timeout | Default: 120 (minutes) | The Connection Timeout system property is used to prevent service outages caused by clients that do not properly close connections. The value can be set in Oracle Enterprise Manager's Server Properties page.<br>**Warning**: Setting to 0 disables the enforcement and client connections can not be closed regardless of how long they are inactive. The system property is not enforced on IP addresses and subjects that are exempt from the quota limit or that have disabled quota enforcement. |

***Table 24–5 (Cont.) Server Parameters***

| Parameter | Value | Notes |
|---|---|---|
| Logging Levels | Default: Error:1 (Severe) | By default, log messages are written to the access.log file only when logging is set to NOTIFICATION:1. To maintain performance, consider keeping the default log level or use WARNING:1 (WARNING) to limit the amount of information written to the access.log file. |

# 25

# Oracle Access Management Performance Tuning

This chapter provides guidelines for tuning and sizing the services that make up an Oracle Access Management 11g Release 11.1.2 installation.



## 25.1 About Oracle Access Management

Oracle Access Management includes a full range of services that provide Web perimeter security functions and Web single sign-on; identity context, authentication and authorization; policy administration; testing; logging; auditing; and more.

Oracle Access Management is a Java Platform, Enterprise Edition (Java EE)-based enterprise-level security application that provides restricted access to confidential information and centralized authentication and authorization services. Many existing access technologies in the Oracle Identity Management stack converge in Oracle Access Management.

Starting with release 11.1.2, Oracle Access Management includes the following "services":

- Oracle Access Management Access Manager (formerly the standalone product named Oracle Access Manager)
- Oracle Access Management Security Token Service (formerly the standalone product named Oracle Secure Token Service)
- Oracle Access Management Identity Federation (formerly the standalone product named Oracle Identity Federation)
- Oracle Access Management Mobile and Social (formerly the standalone product named Oracle Identity Connect)

For more information on administering these services, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

**Note:** Prior to the Oracle Fusion Middleware 11.1.2 release some of the services discussed in this chapter, such as Oracle Identity Federation and Oracle Secure Token Service, were standalone products and tuned individually.

For information on tuning the 11.1.1 standalone versions of these services, see the *Oracle Fusion Middleware Performance and Tuning Guide* in the Oracle Fusion Middleware 11g Release 1 (11.1.1.6.0) documentation library.

## 25.2 Performance Considerations for Oracle Access Management Services

Identifying the areas of your Oracle Access Management environment that may impact performance is the first step in effective performance tuning. This section provides information on some of the common areas to review. Always consult your specific usecase scenarios and performance requirements to determine which configurations are applicable.

Before you begin tuning Oracle Access Management services, review the following sections as well as the recommendations discussed in Chapter 2, "Top Performance Areas":

- Understanding Your Current Environment
- Controlling Network Latency
- Enabling DMS Performance Instrumentation

### 25.2.1 Understanding Your Current Environment

Before tuning Access Management services consider the tuning recommendations described in Table 25–1:

***Table 25–1 Understanding Your Current Environment: Tuning Considerations***

| Tuning Consideration | Description |
|---|---|
| Number of Users | Understanding the overall user population size; group, membership and attribute counts; data types, and configuration parameters of the LDAP and database is essential. See Performance Planning for more information on using population data to improve performance. |
| Daily Activity Usage | **Access Manager**: It is important to know how many users are active during a 24-hour period and the expected traffic. Spikes in usage may require additional tuning to avoid performance issues. See Monitoring Oracle Fusion Middleware for more information on collecting performance data.<br><br>**Identity Federation**: It is important to know how many Federated SSO requests are processed in a 24-hour period and the expected traffic. Spikes in usage may require additional tuning to avoid performance issues |
| Hardware Resources and Topology | Like any application deployed for interactive use in a demanding environment, proper server sizing and configuration is critical for acceptable performance. Ensuring that your hardware is sufficient to prevent bottlenecks is a key factor in performance tuning. See Securing Sufficient Hardware Resources for more information on optimizing hardware resources. |

*Table 25–1 (Cont.) Understanding Your Current Environment: Tuning Considerations*

| Tuning Consideration | Description |
|---|---|
| Partners and Protocols | When tuning Identity Federation, knowing which partners are configured, how those partners are modeled and the federation protocol used are important considerations. Specifically you should understand how many partners this instance has and what protection policies are assigned to them. |
| Protected Applications | Knowing which applications are being protecting and how that protection is modeled is an important consideration when tuning. Specifically you should understand how the applications are being protected: using Webgates (10g,11g, or 11gPS1); mod_osso; custom AccessGates; or a combination. |
| JVM and Garbage Collection | Optimal performance of the Access Management services depends on correctly tuning JVM heap sizes and garbage collection. See Configuring Garbage Collection and Specifying Heap Size Values in Tuning Java Virtual Machines (JVMs)for more information.<br>**NOTE**: When uploading large Plugins or CRLs (10MB+) through the OAM Console UI, you need to ensure that the OAM Server heap size is optimally tuned to overcome OutOfMemory issues.<br>For example, increase the `-Xmx` and `XX:MaxPermSize` if the following error message is seen in the OAM logs:<br>`javax.management.RuntimeErrorException: GC overhead limit exceeded`<br>Use Parallel, Concurrent Mark and Sweep GC modes with the JVM running in the Server Mode. In addition, Oracle reccomends to set the Heap size to a large value and use the same values for Minimum and Maximum `(-Xms=-Xmx)` . |

## 25.2.2 Controlling Network Latency

The performance of the overall network is a major factor in the performance of the system. A reduction in network latency can improve network performance.

To control network latency, consider the following:

- Keep database repositories close to the OAM servers. Installing OAM servers on a remote server may cause significant latency. Latency between the application tier and the database tier should be 5ms or less to maintain optimal performance.
- Add an SSL accelerator or load balancer outside of the Oracle Access Manager system to improve the performance of your network.
- Deploying a load balancer in front of the Web servers or application servers is a best practice for increasing availability and performance of Web-based applications, including Oracle Access Manager. However, load balancers are not recommended between the Oracle Access Manager components themselves.
- Place the Access Manager Servers closer to client applications than to the directory.

  During normal operations there can be a considerable amount of traffic between Webgates and Access Manager Servers. Locating these managed servers closer to the applications can reduce the latency between devices in high-traffic parts of the network.

Access Manager provides keep alive, failover, and failback functionality to handle LDAP and network outages, replication, and related activities. The built-in features of Oracle Access Manager are often the same or better than similar features provided by a load balancer.

**Note:** In addition to ensure fast failover, tune the settings for fast failover. The defaults rely on the OS TCP/IP settings which must be tuned for the OS on which the Webgate is running.

You may use Load Balancers to manage the Access Manager server communication information for OAP (Oracle Access Protocol) by virtualizing it. The benefits of using a Load Balancer between Webgates and Servers should be measured against the following constraining requirements:

- OAP connections are persistent and need to be kept open for a configurable duration even while idle.
- The Webgates need to be configured to recycle their connections proactively prior to the Load Balancer terminating the connections, unless the Load Balancer is capable of sending TCP resets to both the Webgate and the server ensuring clean connection cleanup.
- The Load Balancer should distribute the OAP connection uniformly across the active Access Manager Servers for each WG (distributing the OAP connections according the source IP), otherwise a load imbalance may occur.

**Caution:** If the above constraining requirements are not met, you can negatively impact the performance of Access Manager resulting in outages.

Ensure that the LDAP timeout under load are negligible. This requires ensuring that LDAP Server is appropriately patched and load testing be performed to simulate OAM LDAP queries (bind, user/group lookup, search queries). LDAP timeouts under load increases OAM Server SSO latencies and increase the risk of an OAM server outage.

Temporary latency blips (for example, increase in LDAP query latency, server processing due to increased Coherence latency) results in increased Webgate response times. If the Web Tier does not have adequate capacity to handle the incoming user requests (through queuing or throttling) especially during peak load, you may run into a situation where the entire Web Tier is blocked and unable to accept new requests.This results in end users not being able to login to access business application.

### 25.2.3 Enabling DMS Performance Instrumentation

For performance tuning purposes, consider enabling Dynamic Monitoring Service (DMS) performance instrumentation which can tell you the latency and throughput of functional and operational metrics. DMS can identify components that are either processing a heavier load or taking longer than usual to service requests. See Viewing DMS Metrics for more information on determining the overall time to process calls to various components.

**Note:** If you are using Enterprise Manager Grid Control, create Dashboard Reports based on the OAM Metrics of most interest, which can then be emailed on a regular schedule.

## 25.3 Tuning Oracle Access Management Access Manager

Oracle Access Management Access Manager (Access Manager) is an enterprise level solution that centralizes critical access control services to provide an integrated solution that delivers authentication, authorization, Web single sign-on, policy administration and enforcement, agent management, session control, systems monitoring, reporting, logging, and auditing.

For more information on using Access Manager, see "Introduction to Oracle Access Management Access Manager" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

- Basic Tuning Considerations for Access Manager
- Advanced Tuning Considerations Access Manager
- Specific Use Cases That Require Additional Tuning for Access Manager

### 25.3.1 Basic Tuning Considerations for Access Manager

Depending on your Access Manager usage and performance issues, you may consider tuning the following basic parameters. See Top Performance Areas for additional tuning considerations.

- Tuning the Web Tier
- Managing Policy Components
- Tuning the Data Tier Connections

#### 25.3.1.1 Tuning the Web Tier

Tuning your Web application's server is essential to maintaining optimal performance for Access Manager. This section describes tuning configurations for the following:

- Tuning the Oracle HTTP Server
- Tuning Access Manager Webgate

##### 25.3.1.1.1 Tuning the Oracle HTTP Server

Access Manager Webgate is typically installed on the Oracle HTTP Server. To maximize Access Manager performance, review your use case scenarios to determine the best way to tune the HTTP Server.

At a minimum, consider tuning the following Oracle HTTP Server parameters shown in Table 25–2 in the `httpd.conf` file:

***Table 25–2 Oracle HTTP Server Tuning Parameters and Descriptions for Webgate***

| Parameter | Description |
|---|---|
| `MaxKeepAliveRequests` | A value of zero in the `MaxKeepAliveRequests` directive means there is no limit on the number of connections, which are kept alive expecting subsequent client requests. |
| `Timeout` | The total amount of time it takes to receive a GET request. |

*Table 25–2 (Cont.) Oracle HTTP Server Tuning Parameters and Descriptions for Webgate*

| Parameter | Description |
|---|---|
| KeepAliveTimeout | The number of seconds HTTP Server will wait for a subsequent request before closing the connection. Once a request has been received, the timeout value specified by the Timeout directive applies. See above.<br>Setting KeepAliveTimeout to a high value may cause performance problems in heavily loaded servers. The higher the timeout, the more server processes will be kept occupied waiting on connections with idle clients. |
| If <IfModule mpm_worker_module> has been configured, then consider tuning the following: | |
| StartServer | |
| ServerLimit | |
| MaxClients | |
| ThreadsPerChild | |
| MaxRequestsPerChild | |
| AcceptMutex | |
| LockFile | Consider modifying the OHS LockFile directive to a location on the local disk and not to a shared drive. This will help in avoiding known locking issues on the Oracle HTTP Server Reserve proxy as well as the Oracle HTTP Server Webgate server. |

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

#### 25.3.1.1.2 Tuning Access Manager Webgate

Webgate is an out-of-the-box access client for Access Manager.This Web Server access client intercepts HTTP requests for Web resources and forwards them to the Access Manager Server. Webgates for various Web Servers are shipped with Access Manager. To maximize performance, consider tuning the Webgate connections to the Access Manager server.

Consider tuning the following parameters to increase the number of connections from the Webgate Server to the Access Manager servers. Adding more connections enables the servers to process more concurrent requests.

| Parameter | Description |
|---|---|
| Max Connections | Maximum number of connections that this Access Manager Agent can establish with all the Access Manager Servers. |
| Maximum Number Of Connections | Maximum number of connections that the Access ManagerAgent can establish with a specified Access Manager Server. |

For more information on setting these parameters, see "Registering Agents and Applications" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

#### 25.3.1.2 Managing Policy Components

In order to limit the Access Manager processing overhead, all resources that do not require security should be modeled as excluded resources as opposed to unprotected resources. Modeling these resources as excluded resources can substantially help with ADF Applications. Excluded resources use a one-time interaction between the Webgate and the Access Manager Server as opposed to a per request interaction for unprotected resources.

For more information, see "Managing Shared Policy Components" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

To design authentication policies for optimal performance, do the following:

- Get an inventory of all attributes you want for `authZ` and pre-fetch them at `AuthN` time.
- Combine attributes in the supplementary list to reduce `AuthN` time LDAP load.

Note the following:

1. Change all OAM policy responses for userid return from `$user.attr.uid` to `$user.userid`.This is because the latter is computed at login time as opposed to the former which is computed onDemand during authorization

   `OAM_REMOTE_USER` is populated by default.

2. To design authorization policies for optimal performance, do the following:

   - Use `$session namespace`. Attributes used for authroization must be retrieved and stored in the user's OAM session during login. This ensures that the `authZ` latency is constant to make OAM responsive thereby improving the user experience.

     For example, modify `ismemberof`, `loa` and any other attribute related policy response to get value at authentication time instead of `authZ` time.

     ```
     [Authentication Policies]
     ismemberof -> SESSION -> $user.attr.ismemberof
     loa -> SESSION -> $user.attr.loa
     uid ->SESSION -> $user.userid

      [Authorization Policies]
     Responses:
     uid: $user.userid
     ismemberof: $session.attr.ismemberof
     loa: $session.attr.cmsRoles
     ```

   - For Authorization policies involving attributes, store and use attributes in the `$session namespace` instead of query them on-the-fly by using the `$user.attr namespace`.
   - Use group based policies instead of explicitly listing users.

#### 25.3.1.3 Tuning the Data Tier Connections

LDAP stores are accessed by connection pools maintained by Access Manager. Identity store definitions contain the exposed pool parameters. Middleware Control and the DMS Spy Servlet can expose per-operation counts and latency which can be used to identify bottlenecks. Consider specifying an explicit time-out value (default=unlimited) and ensure that the initial and maximum number of connections in the pool are appropriate for the deployment.

For more information, see "Managing User Identity Stores" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

## 25.3.2 Advanced Tuning Considerations Access Manager

The following Access Manager tuning considerations are provided as a guide. Always consult your own use case scenarios to determine if these configurations should be used in your deployment.

- Tuning Oracle Coherence
- Setting the Java Message Bean Pool Size
- Tuning the Server Cache
- Tuning Webgate Caches
- Changing Request Cache Type
- Tuning Authentication Plug-Ins

### 25.3.2.1 Tuning Oracle Coherence

Oracle Access Manager uses Oracle Coherence to replicate session states within a distributed installation. Coherence is used to communicate state changes between the Oracle Access Manager Console and Access Manager Servers.

Oracle Coherence recommends that you configure your operating system (OS) to allow for larger buffers. Consider increasing the buffer to at least 2 MB.

`coherence.distributed.threads` in `oam-confgi.xml` must be set to a minimum of 16.

Coherence monitoring is current disabled by default in the OAM Server. To enable Remote Monitoring of Coherence over JMX, do the following:

1. In the `oam-config.xml` file of the server, locate the configuration element at the path `/DeployedComponent/Server/NGAMServer/Profile/CoherenceConfiguration/ServerTypeSettings/AdminServer`. Verify that a child element Management exists. If it does not exist, create one as follows:

   ```
   <Setting Name="Management" Type="htf:map">
   <Setting Name="Key"
   Type="xsd:string">oam.coherence.management</Setting>
   <Setting Name="Value" Type="xsd:string">all</Setting>
   </Setting>
   ```

   Remove any other "Management" element that may exist in the configuration file.

2. Locate the element "RemoteManagement" in the configuration file. It will exist as follows:

   ```
   <Setting Name="RemoteManagement" Type="htf:map">
   <Setting Name="Key"
   Type="xsd:string">oam.coherence.management.remote</Setting>
   <Setting Name="Value"
   Type="xsd:boolean">false</Setting>
   </Setting>
   ```

   Change the value of the Value element to true.

3. For the changes to take effect, you can increment the value of the "Version" element of the configuration or restart the OAM Servers.

#### 25.3.2.2 Setting the Java Message Bean Pool Size

By default, the Access Manager Proxy is set to handle 100 concurrent Webgate requests. If necessary, consider adjusting the pool settings to reflect the maximum Webgate request load for the deployment. This is achieved by setting the `max-beans-in-free-pool` element to an appropriate value. This deployment configuration is available in the Weblogic Server Administration Console. For more information, see Configuring Connection Pools.

To choose the appropriate value for the `max-beans-in-free-pool`, calculate it based on the Web Tier settings discussed in Tuning the Web Tier. This value should be greater than the `Max Number of connections` (in Webgate) multiplied by the `ServerLimit` (in Oracle HTTP Server) multiplied by the `Number of Webgates`.

#### 25.3.2.3 Tuning the Server Cache

The following server caches can be tuned to improve Access Manager performance:

##### 25.3.2.3.1 Tuning Identity Store Cache

Authorization policy administration allows authoring of grants to users or groups. Administrators can search within specific identity stores, selecting certain users or groups and granting or denying them access. Search results provide canonical identifiers for users and groups such that those values are stored as principals of the Identity Constraint component of Access Manager Authorization policy. The console displays the names and the Identity Store of origin.

To maximize performance, review configuration settings of the following Identity Store caches:

- Group Membership Cache

  The Group Membership cache stores indirect membership data which is essentially a group's membership in another group. The number of entries and entry time-to-live are configurable parameters. The cache should be tuned if your deployment includes groups that will be checked against or exported as responses, such as groups that are set in identity constraints, for example.

  **CAUTION**: The Group Membership Cache is populated by a recursive search of the entire LDAP tree of nested groups without any loop detection. Consider disabling this cache if you are experiencing degraded Access Manager Server performance.

- User Attribute Cache

  User Attributes, once fetched, are always cached. Pre-fetching of attributes during authentication is controlled by specifying the attribute list in the SUPPLEMENTAL_RETURN_ATTRIBUTES parameter value of the Identity Store.

  Supplemental attribute return values are useful when you do not require the user to make a list selection for the attributes, yet you want those attributes values, as determined by the current row, to participate in the update.

  **Note:** All LDAP Attribute Condition used in Authz Policy must be retrieved during login and be cached. This improves authz latency and throughout while reducing the burden on the LDAP tier.

#### 25.3.2.4 Tuning Webgate Caches

Webgate caches information on authentication and on whether or not a resource is protected. Webgate cache tuning sets the total number of unique URLs expected over the timeout interval. Default is 0 URLs, but this means that the cache is not automatically updated and is flushed only when the administrator manually updates the cache. While this is a good option for performance in some scenarios, it may not apply to your individual use cases.

This section provides the following topics:

- Introducing Webgate Caches
- Reducing Network Traffic Between Components
- Changing the Webgate Polling Frequency

For more information, see "Reviewing OAM Agent Metrics" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

**25.3.2.4.1 Introducing Webgate Caches** Webgate caches various information related to resources, authentications and authorizations to improve performance. It uses the cached information to avoid trips to 11g Server for requesting same information. Table 25–3 are the caches used by Webgate to maintain this information.

***Table 25–3 Webgate Cache Types***

| Cache Type | Description |
|---|---|
| Resource to Authentication Scheme | This cache maintains information related to authentication schemes being used.<br>Default: 100000 elements<br>See Also: "Tuning Maximum Cache Elements" and "Tuning Cache Timeout Values". |
| Authentication Scheme | This cache maintains information related to authentication schemes being used.<br>Default: 25 elements<br>Typically Authentication Scheme cache elements require less than 2 Kb of memory per element.<br>See Also: "Tuning Cache Timeout Values". |
| Resource to Authorization Policy<br>***11g Webgate only*** | This cache maintains information related to resources accessed and associated authorization policy.<br>Default: 100000 elements<br>See Also: "Tuning Maximum Cache Elements" and "Tuning Cache Timeout Values". |
| Authorization Result<br>***11g Webgate only*** | This cache maintains information related to authorizations associated with user sessions.<br>Default: 1000 elements<br>See Also: "Tuning Authorization Result Cache" and "Tuning Authorization Result Cache Timeout". |

**See Also:** "About the 11g Webgate Diagnostics Page"

### About the 11g Webgate Diagnostics Page

This page displays useful information related to currently effective Cache configuration parameters. It also displays runtime information about the caches that include information on the number of cached elements, number of hits and misses so far, and current memory usage of individual caches. The page is found at the following URL:

http://*webserver:port*/ohs/modules/webgate.cgi?progid=1

After upgrading Oracle Webgate 10.1.4.3.0 to Bundle Patch 13 (BP13), the output of the Diagnostic page is a blank page.Starting with Bundle Patch 13, the Diagnostic Page is disabled by default.

To enable this page, per webgate registration, add the parameter/value : `enableDiagnosticPage=true` in the list of user parameter of the webgate.With a webgate instance already registered :

- Go to `OAM Console > System Configuration > Access Manager > SSO Agents > OAM Agents` : Search and Select your Webgate 10g profile
- Add in the end of the list of the `"User Defined Parameters" : enableDiagnosticPage=true`.
- Click on Apply : a pop-up window mentions where the new artifacts are located.
- Copy the newly `ObAccessClient.xml` in the OHS configuration instance.
- Restart the OHS instance and check that the Diagnostic Page is displayed.

> **Note:** Changes to Webgate parameters are not reflected on Webgate until the next configuration refresh. For 11g Agents, the default configuration refresh interval is 10 minutes.

### Tuning Maximum Cache Elements

By default, the Resource to Authentication Scheme and Resource to Authorization Policy caches are created to store 100000 elements. Typically, elements of these caches require less than 1 Kb of memory per element. Therefore, with 100000 elements in each of these caches, typical memory requirement for the caches will be 100000 Kb or 100 Mb each.

Considering memory requirements and your deployment, the Web Server being used and number of unique URLs in your application, you might want to increase or decrease the maximum number of elements to be cached.

> **Note:** Increase or decrease the Maximum Cache Elements parameter value as needed. If this is set to a value of -1, all Webgate caches are disabled.

For both 10g and 11g Webgates, you can tune the maximum number of elements to be cached property, by changing the Maximum Cache Elements parameter. Updates to this parameter require a Webgate restart.

#### To tune the maximum number of elements to be cached

1. Locate and open the desired 10g or 11g Webgate registration page in the Oracle Access Management Console.
2. Set the Maximum Cache Elements parameter as desired.
3. Restart Webgate Web server.

### Tuning Authorization Result Cache

By default, the Authorization Result cache is created to store 1000 elements. Authorization Result cache elements store the user session identifier, authorization policy identifier, and associated authorization result including any processed policy

responses. Therefore, Authorization Result cache elements are bulky and generally require more than 2Kb of memory per element.

Considering memory requirements and the number of concurrent user sessions in your deployment, you might want to increase the number of elements to be cached.

#### To tune the number of elements to be cached

1. Locate and open the desired 11g Webgate registration page in the Oracle Access Management Console.
2. In User Defined Parameters, add or update maxAuthorizationResultCacheElems as desired.
3. Restart Webgate Web server.

### Tuning Cache Timeout Values

By default, the following caches are created with a timeout value of 1800 seconds or 30 minutes:

- Resource to Authentication Scheme
- Authentication Scheme
- Resource to Authorization Policy

Elements in these caches are stored with an expiry time that forces these caches to be flushed on expiry.

Considering the frequency of updates to Authentication Schemes, and Authentication and Authorization Policies in your deployment, you might want to increase or decrease the default timeout value.

#### To tune the cache timeout

1. Locate and open the desired 10g or 11g Webgate registration page in the Oracle Access Management Console.
2. Set the Cache Timeout parameter as desired.
3. Restart Webgate Web server.

### Tuning Authorization Result Cache Timeout

By default, the Authorization Result Cache timeout value is set at 15 seconds. Elements in the Authorization Result Cache is stored with an expiry time that forces it to be flushed on expiry. A low timeout value ensures that authorization results are cached for a small amount of time only.

Considering average length of user sessions and frequency with which user sessions are created and destroyed, you might want to change the default timeout value. Unlike other caches and parameters, updates to this parameter do not require Webgate restart. Instead, the updated value is dynamically picked up by 11g Webgate and enforced immediately.

---

**Note:** If authorizationResultCacheTimeout is set to 0, Authorization Cache is disabled.

---

#### To tune the authorization result cache timeout

1. Locate and open the desired 11g Webgate registration page in the Oracle Access Management Console.

2. In User Defined Parameters, add or update authorizationResultCacheTimeout as desired.
3. Restart Webgate Web server.

#### 25.3.2.4.2 Reducing Network Traffic Between Components

The Webgate-to-OAM Server configuration polling reduces the traffic between both the Webgate and OAM Server and the OAM Server and the registered data stores for Oracle Access Manager.

### Process overview: Webgate-to-OAM Server configuration polling

1. When the Webgate is inactive for 60 seconds, it reduces the frequency of polling for its configuration information.

   The polling frequency is determined by the parameter InactiveReconfigPeriod, which is a user-defined parameter that is set in the Webgate configuration page. The value for InactiveReconfigPeriod is specified in minutes. Within ten seconds of resuming activity, the Webgate performs reconfiguration polling once a minute.
2. At startup, the Webgate checks the bootstrap configuration to see if any important parameters have changed.

   This makes the re-initialization process unnecessary in most cases and reduces the transient OAM Server load.
3. Webgate  configurations are cached in the OAM Server.

   The default cache timeout is 59 seconds. This should cause no modifications to the system behavior on non-Apache access clients. The Apache Web server with Webgate avoids unnecessary hits to the directory server. The caching parameters can be set in the Webgate registration page.

   - Max Cache Elements sets the maximum size of the cache (default 9999)
   - Cache Timeout determines the maximum lifetime of any element in the cache (default 59 seconds)

There are two ways to reduce off-time network traffic between both the Webgate and OAM Server and the OAM Server and the database:

- Changing the default configuration cache timeout for Webgate  configurations that are cached in the OAM Server, as described in Step 3.
- Changing Webgate polling frequency for configuration information, as described next.

#### 25.3.2.4.3 Changing the Webgate Polling Frequency

One way to reduce off-time network traffic between both the Webgate and OAM Server and between the OAM Server and the database is to change the Webgate polling frequency using the InactiveReconfigPeriod parameter.

The default is 1 minute. When the Webgate is inactive for more than 60 seconds (for example, when no authentication requests are being processed), it reduces the frequency of polling for its configuration information. Within ten seconds of resuming activity, the Webgate resumes reconfiguration polling once every minute:

- If set to -2, Webgate never polls.
- If set to a value greater than 0 it polls at the specified interval.
- If set to -1 and Webgate is inactive and has been for 1 minute, then Webgate does not poll. Webgate resumes reconfiguration polling when it returns to an active state.

For example, the OAM Server reads the shared secret from the directory at an interval of 10 minutes and this cached value is returned to Webgate. In the idle state the Webgate reads the shared secret from the OAM Server using the InactiveReconfigPeriod value. If this value is not set, the Webgate polls the OAM Server for the shared secret value at an interval of 1 minute even though the updated shared secret value will be returned only after 10 minutes.

#### To change the configuration polling frequency

1. Locate the desired Webgate registration page using instructions in "Searching for a Webgate Registration" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.
2. Add the InactiveReconfigPeriod parameter as a user-defined parameter on the Webgate registration page.
3. Specify the value for InactiveReconfigPeriod in minutes.
4. Apply your changes to the Webgate registration page.

#### 25.3.2.5 Changing Request Cache Type

The default Request Cache type is set to `COOKIE`, which relies on the use of cookies to cache an unauthenticated request state.

Changing the type to `BASIC` can improve performance, but it is important to consider the following: If the server being used for an authentication flow goes down in the middle of that flow, the user's current state in the flow will be lost on their next request as the load balancer sends them to a different server.

Changing the type to `FORM` can improve performance when lengthy URLs are being accessed.

#### 25.3.2.6 Tuning Authentication Plug-Ins

Authentication plug-ins can affect performance. When you develop customizations for Access Manager, consider the following to minimize performance impact:

- Evaluate the sequence in which actions are executed
- Minimize the plug-in footprint and external dependencies whenever possible

### 25.3.3 Specific Use Cases That Require Additional Tuning for Access Manager

This section describes some specific use cases that require specialized tuning, in addition to the Basic Tuning Considerations for Access Manager.

#### 25.3.3.1 Managing Access Manager Sessions

By default, there can only be a maximum of 8 concurrent sessions for a given user ID. It is possible to raise this limit, but it is important to note that as the limit increases the security value of the feature is eroded, and ultimately disappears. Further, there is a performance cost associated with the feature, which increases with the limit. Therefore, if there is a need to have more than 20 concurrent user sessions, then consider disabling this feature by setting the limit to 0.

#### 25.3.3.2 Audit Settings

OAM tends to generate a lot of audit information. During peak business hours, OAM generates audit information at a rate that is faster than the rate at which the OPSS AuditLoader can move the information to the Audit Database.

Given that SSO is a security service, it is recommend to set the Audit Filter to a value of `MEDIUM` or `ALL`. Also, ensure that the Audit BusStop directory has no max size limit (`maxDirSize=0`) to avoid zero data losses. In addition, monitor and confirm that the Audit data is constantly being moved to the Audit Database even if the AuditLoader falls behind during peak business hours.

#### 25.3.3.3 Managing Monitor Account

Enterprises use automated monitors to measure end user latency and generate alerts when thresholds are exceeded.

Oracle recommends the following best practices:

1. Monitors should logout when their work is done. This ensures that sessions do not pile up in memory.
2. Monitors should not use the same user credential. This ensures that a single user does not create a very large number of sessions in a short amount of time.
3. Prune monitor sessions periodically. This can be done through the OAM Console or by writing a program using the ASDK. It also ensures that you do not have to set the maximum number of sessions to a very large value to accommodate monitors.
4. Refrain from running monitors very frequently when problems are seen.

   Typically, monitors are set to run very frequently when an exception condition is noted (for example, when login latencies exceed the threshold). This has the effect of putting additional load on the system especially if this happens under peak load and this increases the risk of a catastrophic failure.

#### 25.3.3.4 Kerberos Latency Issues

Kerberos authentication, by default, uses the UDP protocol. However, UDP does not perform well when the connection between the OAM Server and Kerberos Server has to span subnets or the packet loss increases during business hours. As a result, it recommended that Kerberos be configured to use TCP instead of UDP.

This can be done by setting `udp_preference_limit=1` in the `/etc/krb5.conf` file.

## 25.4 Tuning Oracle Access Management Identity Federation

Oracle Access Management Identity Federation (Identity Federation) 11gR2 is an identity federation server built into the Oracle Access Manager server. All configuration is performed in Oracle Access Manager; unlike the standalone 11gR1 version. Identity Federation provides a self-contained and flexible multi-protocol federation server that can be rapidly deployed with existing identity and access management systems. It enables you to securely share identities across vendors, customers, and business partners without the increased costs of managing, maintaining, and administering additional identities and credentials.

For more information on administering Oracle Access Management Identity Federation, see "Introduction to Identity Federation in Oracle Access Management" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

### 25.4.1 Basic Tuning Considerations for Identity Federation

The following sections describe basic tuning configurations that you should also consider while tuning Identity Federation:

- Tuning the Load Balancer and HTTP Server
- Tuning SOAP Connections
- Tuning the Data Tier Connections

#### 25.4.1.1 Tuning the Load Balancer and HTTP Server

As of Oracle Fusion Middleware Release 11gR2, some of the features of Identity Federation are embedded in Access Manager. To optimize Identity Federation performance, follow the Load Balancer and HTTP Server tuning guidelines discussed in Tuning the Web Tier for Access Manager.

#### 25.4.1.2 Tuning SOAP Connections

Identity Federation uses the Simple Object Access Protocol (SOAP) to send Security Assertion Markup Language (SAML) requests and to receive SAML responses. To optimize performance, configure the following SOAP connections:

- Total maximum number of SOAP connections that Identity Federation and Security Token Service can open at the same time
- Maximum number of SOAP connections that Identity Federation and Security Token Service can open at the same time to a given remote server

#### 25.4.1.3 Tuning the Data Tier Connections

LDAP stores are accessed by connection pools. Identity store definitions contain the exposed pool parameters. As discussed in Section 25.3.1.3, Middleware Control and the DMS Spy Servlet can expose per-operation counts and latency. Identity Federation uses an RDBMS to store session and runtime data. The server uses a caching mechanism to improve performance at run time. This enables the server to keep a reference to recently used objects in memory to avoid read access to the database. In addition there is an asynchronous write and delete mechanism to the RDBMS.

> **Note:** The following parameters typically do not need to be changed. Review the descriptions, however, to determine if an adjustment could improve performance for your deployment.

To optimize RDBMS session caching and asynchronous writes, configure the parameters as described in Table 25–4:

***Table 25–4 Asynchronous Write Settings***

| Parameter | Description |
|---|---|
| `rdbmsasynchronousmanagerinterval` | Execution interval for the asynchronous thread manager |
| `rdbmsasynchronousmanagersleep` | Sleep interval for the asynchronous thread manager, to check if execution should occur |
| `rdbmsasynchronousqueuesize` | Size of the queue containing RDBMS operations of the same type (create session, create artifact…)<br>**NOTE:** It is important to size the `rdbmsasynchronousqueuesize` correctly. If it is made too large, it can cause a lag in the asynchronous write to the database and may cause SSO operation to fail. |

*Table 25–4 (Cont.) Asynchronous Write Settings*

| Parameter | Description |
|---|---|
| rdbmsasynchronousqueuesleep | Sleep time before the calling thread can retry to add an operation to a queue, in case the queue is full |
| rdbmsasynchronousqueueretries | Number of retries when trying to add an operation to the queue |
| rdbmsasynchronousthreadcore | Number of default threads in the RDBMS thread executor module for RDBMS asynchronous operations |
| rdbmsasynchronousthreadkeepalive | Maximum amount of time to keep the extra threads in the RDBMS thread executor module for RDBMS asynchronous operation |
| rdbmsasynchronousthreadmax | Maximum number of threads in the RDBMS thread executor module for RDBMS asynchronous operation<br>rdbmsasynchronousthreadmax should be adjusted to handle the maximum system load based on the size of your system. |
| rdbmsasynchronousthreadpolicy | Thread policy of the RDBMS thread executor module for RDBMS asynchronous operation |
| rdbmsasynchronousthreadqueuesize | Size of the thread queue of the RDBMS thread executor module for RDBMS asynchronous operation |

Table 25–5 describes the RDBMS memory cache settings for artifact and transient cache:

*Table 25–5 Cache Settings*

| Parameter | Description |
|---|---|
| **RDBMS Artifact memory cache settings, used in conjunction of the RDBMS asynchronous module:** | |
| artifactrdbmscachetimeout | Time to live in the memory cache |
| artifactrdbmsretries | Maximum number of time to retry to locate an entry in RDBMS before returning a failure |
| artifactrdbmssleep | Sleeping time between retrying lookup operations |
| **RDBMS Memory cache settings (except for Artifact):** | |
| transientrdbmscachesize | Size of the cache |
| transientrdbmscachetimeout | Time to live for the objects in the cache, before being invalid and thus forcing an RDBMS lookup operation when an object is searched |
| Interval for the RDBMS cleanup thread | Indicates the interval of sleep of the thread removes expired entries from OIF DB tables |

## 25.4.2 Advanced Tuning Considerations for Identity Federation

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 Identity Federation deployment.

#### 25.4.2.1 Tuning Oracle Coherence

Identity Federation, as part of Access Manager 11gR2, uses Oracle Coherence to replicate session states within a distributed installation. See Tuning Oracle Coherence for more information.

#### 25.4.2.2 Tuning Identity Store

Identity Federation, as part of Access Manager 11.1.2.0.0, will benefit from tuning the identity store as discussed in Tuning the Server Cache.

#### 25.4.2.3 Tuning Protocol Binding

This section describes the protocol binding options:

- XML Digital Signatures

  Identity Federation relies on XML Digital Signatures to ensure the authenticity of messages and that messages are not tampered with.

  When possible, sign the Assertion and/or the Response to prevent any modifications. When no XML Digital Signature is present on the message, the audited message that is archived does not contain any data that proves the authenticity and integrity of the message.

  Configuring Identity Federation or Security Token Service to not sign Assertion and/or Response may be appropriate if:

  - Performance must be improved
  - SSL with SSL authentication is enabled for SOAP communications
  - Disabling XML Digital Signatures is compliant with company security regulations

- XML Encryption

Federated Single Sign-On allows the use of token and element level encryption to provide confidentiality to the message exchange. Disabling use of encryption improves the latency and throughput of Identity Federation.

#### 25.4.2.4 Tuning the Browser POST and Artifact Single Sign-On Profiles

There are two Single Sign-On profiles defined by the SAML specifications:

- POST Profile

  In the POST profile, the Assertion transits through the user's browser, therefore the Assertion and/or the Response must be signed to ensure that the content has not been modified.

- Artifact Profile

  In the Artifact profile, the Identity Provider creates a random identifier referencing the Assertion in the IdP's local store. (The Assertion is provided directly from the Identity Provider to the Service Provider.) That identifier is carried by the user's browser and presented to the Service Provider that contacts the Identity Provider to de-reference the identifier and retrieve the corresponding Assertion.

  If the SOAP connection made from the SP to the IdP is encrypted using the SSL protocol with an SSL Server Certificate, then the SP authenticates the IdP and the content of the communication has not been tampered with: in this case, the transport layer is providing the authenticity and the integrity of the message, and the XML Digital Signature on the SAML Response and Assertion can be optional.

If no XML Digital Signature is present on the message, then the audited message that is archived does not contain any data that proves the authenticity and integrity of the message.

Since the Artifact profile involves an additional round trip between the Service Provider and the Identity Provider, you may be able to improve performance by avoiding use of the Artifact profile.

### 25.4.3 Specific Use Cases That Require Additional Tuning for Identity Federation

This section describes some specific use cases that may benefit from additional tuning.

#### 25.4.3.1 Message Signing versus Token Signing

Message exchange between the Service and Identity providers may be signed. Message signature provide additional security when the request/response transits numerous intermediaries. Disabling message signatures can improve performance but this should be  done only when the security risk of doing so is mitigated by other security mechanisms

## 25.5 Tuning Oracle Access Management Security Token Service

Oracle Access Management Security Token Service (Security Token Service) provides a centralized mechanism to broker trust between applications and web services by enabling seamless propagation of identities and security context.

For more information on administering Security Token Service, see "Introduction to Oracle Access Management Secure Token Service" in *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

### 25.5.1 Basic Tuning Considerations for Security Token Service

The following sections describe basic tuning configurations that you should also consider while tuning Security Token Service:

- Tuning the Load Balancer and HTTP Server
- Tuning SOAP Connections
- Tuning the Data Tier Connections

#### 25.5.1.1 Tuning the Load Balancer and HTTP Server

To optimize Security Token Service performance, follow the Load Balancer and HTTP Server tuning guidelines discussed in Tuning the Web Tier for Access Manager.

#### 25.5.1.2 Tuning SOAP Connections

Security Token Service uses the Simple Object Access Protocol (SOAP) to send Security Assertion Markup Language (SAML) requests and to receive SAML responses. To optimize performance, configure the following SOAP connections:

- Total maximum number of SOAP connections that can open at the same time
- Maximum number of SOAP connections that can open at the same time to a given remote server

#### 25.5.1.3 Tuning the Data Tier Connections

Security Token Service uses an RDBMS to store runtime data. The server uses a caching mechanism to improve performance at run time. This enables the server to keep a reference to recently used objects in memory to avoid read access to the database. In addition there is an asynchronous write and delete mechanism to the RDBMS. See Section 25.4.1.3, and review the tuning parameters discussed in Table 25–4 and Table 25–5 as these parameters should also be set for Security Token Service.

In addition, because the LDAP connections are made from Security Token Service when LDAP credential validation is enabled in a validation template in Security Token Service, the connections to that LDAP instance should be tuned with the following parameters:

- Setting the LDAP Inactivity setting which tells Security Token Service how long an LDAP connection should be kept in a pool before being removed due to inactivity.

  Over time, the LDAP server may close some connections due to a long inactivity period, and if left unchecked, this can result in errors and may impact performance.

- Setting the LDAP Read Timeout Setting. Sometimes the LDAP server can become unresponsive, causing the thread/user to wait for a response or an error.

  To avoid waiting too long for an error when the server is not responding, Security Token Service sets a read timeout property on the LDAP connection. If the LDAP server does not respond before the read timeout period, an error is generated. Security Token Service closes the connection, open a new one and re-issue the LDAP command.

- Setting the High Availability (HA) LDAP Flag.

  When integrated with LDAP Servers that are deployed in HA mode, STS must configured to indicate that the LDAP Servers are in HA mode.

### 25.5.2 Advanced Tuning Considerations for Security Token Service

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 Security Token Service deployment.

#### 25.5.2.1 Tuning the WS-Security Policy

To optimize Security Token Service performance, consider following the recommendations below when configuring your WS-Security Policy:

- Optimal use of Integrity, Confidentiality and RequiredElements assertion
- Optimal use of security binding properties
- Use TransportBinding  over SymmetricBinding, which in turn should be considered before AsymmetricBinding
- Avoid encrypting the token for the WS Provider

## 25.6 Tuning Oracle Access Management Mobile and Social

Oracle Access Management Mobile and Social (Mobile and Social) is a new intermediary between a user seeking access to protected resources, and the back-end Identity and Access Management services that protect the resources. Mobile and Social provides simplified client libraries that allow developers to quickly add feature-rich

authentication, authorization, and identity capabilities to registered applications. On the back-end, the Mobile and Social pluggable architecture lets system administrators add, modify, and remove Identity and Access Management services without having to update user installed software

## 25.6.1 Basic Tuning Considerations for Mobile and Social

The following sections describe basic tuning configurations that you should also consider while tuning Mobile and Social:

- Tuning the Access Management Authentication Service Provider
- Tuning the User Profile Service Provider

### 25.6.1.1 Tuning the Access Management Authentication Service Provider

Mobile and Social has an out-of-the-box Oracle Access Management Authentication Service Provider which connects to Oracle Access Management server using Access Manager SDK components. To optimize Mobile and Social, consider tuning Access Manager as described in Section 25.3, "Tuning Oracle Access Management Access Manager".

In addition to tuning the Access Manager configuration parameters, there is one configuration parameter that should be tuned in Mobile and Social:

***Table 25–6 Mobile and Social Tuning Parameters***

| Parameter | Description |
|---|---|
| `OAM_SERVER_x_MAX_CONN` | Use the following steps to configure the maximum number of connections provided for the Access Management server:<br>1. From the Access Manager 11g R2 console, click **System Configuration** tab and select **Mobile and Social** on the left panel<br>2. Under "Authentication Service Provider", select the target Access Manager service provider<br>3. Change the value for `OAM_SERVER_x_MAX_CONN` properly for your performance requirements.<br>4. Save the change.<br>**NOTE**: This parameter should be set to be the same value as defined for the "Max Connection for webgate agent" in Access Manager. If different values are provided then the setting in Access Manager server will take precedence. |

### 25.6.1.2 Tuning the User Profile Service Provider

The User Profile Service in Mobile and Social depends on IDS/libOVD to connect to the user repository. There are two IDS/libOVD configuration parameters that can be tuned for the production deployment as described below. These parameters can be changed via Mobile and Social Administration Console.

***Table 25–7 User Profile Service Provider Tuning Parameters***

| Parameter | Description |
|---|---|
| Connection Pool Initial Size | Category: LDAP Adapter Properties<br>Default: 5<br>Recommendation: The default value an be used. |

*Table 25–7 (Cont.) User Profile Service Provider Tuning Parameters*

| Parameter | Description |
|---|---|
| Connection Pool Maximum Size | Category: LDAP Adapter Properties<br>Default: 10<br>Recommendation: Tune the size of the LDAP connection pool in Oracle Virtual Directory LDAP Adapter to be at least as high as the total number of Threads configured in the Oracle Virtual Directory Listeners that actively use the LDAP Adapter. |

For more information about libOVD configuration parameters, see Chapter 24, "Oracle Virtual Directory Performance Tuning".

# 26

# Oracle Identity Manager Performance Tuning

This chapter provides guidelines for tuning and sizing specific to Oracle Identity Manager (OIM). It contains these topics:

- Section 26.1, "About Oracle Identity Manager"
- Section 26.2, "Monitoring Oracle Identity Manager Performance"
- Section 26.3, "Basic Tuning Considerations"
- Section 26.4, "Advanced Tuning Considerations"

> **Note:** As with any enterprise class business application, there is no simple procedure for tuning that works for all systems. The tuning sections in this chapter provide (in some cases) sample configurations and outline the principles for tuning Oracle Identity Manager. Consider your own use case scenarios to determine which settings are appropriate.

## 26.1 About Oracle Identity Manager

Oracle Identity Manager (OIM) provides operational and business efficiency through centralized administration and complete automation of identity and user provisioning events across the enterprise, as well as extranet applications.

For more information on using Oracle Identity Manager, see *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

## 26.2 Monitoring Oracle Identity Manager Performance

To identify performance bottlenecks, you can monitor real-time performance metrics for the Oracle Identity Manager database. For more information on how to monitor your Oracle Fusion Middleware components, see Chapter 4, "Monitoring Oracle Fusion Middleware".

For Oracle Identity Manager it is recommended that you perform the following at regular intervals:

- Monitor real-time performance by using a performance-monitoring tool such as Oracle Enterprise Manager console or Automatic Workload Repository (AWR) in Oracle Database 11*g*.

**Note:** You can use Oracle Enterprise Manager 11g Fusion Middleware Control to monitor Oracle Identity Manager. To do so:

1. Under Identity Management, select **Oracle Identity Manager** to go to the home page. On the Home page, you can monitor Oracle Identity Manager.
2. From the Oracle Identity Manager menu, select **Performance** to view performance metrics.

- Collect routine statistics and report by using Oracle Database Enterprise Manager (EM), which is available in Oracle Database as a standard offering.
  - Routine Statistics Gathering

    Routine statistics gathering can be taken care by the 'Automated Maintenance Tasks', which is available in the following navigation path in Oracle Database:

    **Oracle EM**, the **Server** tab, **Query Optimizer**, **Manage Optimizer Statistics**, the **Automated Maintenance Tasks** link
  - Reporting requirements of statistics through Oracle Database 11g EM

    To report on the state of the currently gathered statistics, EM provides a reporting interface in the following navigation path:

    **Oracle EM**, the **Server** tab, **Query Optimizer**, **Manage Optimizer Statistics**, the **Object Statistics** link

    This interface can be used for the reporting purpose for All Objects (of the Schema or even the Object of choice), which have Stale, Missing, or Locked states or are already analyzed.
- Collect complete schema statistics upon implementation of Oracle Identity Manager.

  Update schema statistics regularly, so that the Cost-Based Optimizer (CBO) can access the latest statistics. You must consider complete schema or table statistics on mass data change events such as bulkload of users or accounts, import of a new connector, a huge reconciliation run from a new target system, or use of an archival utility.

  This helps the CBO determine an efficient query execution plan that is based on the current state of data. The following is a sample SQL command to collect database statistics on a regular basis:

**See Also:** Gathering routine statistics and reporting can be done by performing the automated maintenance tasks available in Oracle Database 11g. See *Oracle Database Performance Tuning Guide 11g Release 1 (11.1)* for details.

```
DBMS_STATS.GATHER_SCHEMA_STATS(OWNNAME=> schema_owner,
Exec dbms_stats.gather_schema_stats(OWNNAME=> 'OIM_OIM',ESTIMATE_PERCENT=>DBMS_
STATS.AUTO_SAMPLE_SIZE,options=>'GATHER AUTO',degree => 8,cascade=>TRUE);
```

- Look for relevant recommendations provided in advisory sections in the Automatic Database Diagnostic Monitor (ADDM) or Automatic Workload Repository (AWR) report, and adjust the instance configuration parameters according to the recommended settings. This is specially required after importing a new connector and completing a round of reconciliation from a new target

system so that you can identify the need of any new indexes according to your matching rules.

## 26.3 Basic Tuning Considerations

Depending on your Oracle Identity Manager usage and performance issues, you may consider tuning the following basic parameters. See Chapter 2, "Top Performance Areas" for additional tuning considerations.

- Tuning and Managing Application Cache
- Tuning the Application Server for Oracle Identity Manager
- Tuning Database Parameters for Oracle Identity Manager
- Tuning Oracle Internet Directory
- Tuning Application Module (AM) for User Interface

### 26.3.1 Tuning and Managing Application Cache

Oracle Identity Manager allows caching of metadata, which reduces DB activities. This results in reduced network load and improved performance.

By default, caching for most of the configurations are disabled (set to false) so that the configuration changes are reflected immediately without having to restart the application servers in the development environments.

The following sections provide some recommended cache values for tuning Oracle Identity Manager:

- Tuning Oracle Identity Manager Cache
- Purging the Cache

#### 26.3.1.1 Tuning Oracle Identity Manager Cache

Caching is configured in the `/db/oim-config.xml` configuration file, which is located in MDS where Oracle Identity Manager stores the configuration. You can use Oracle Enterprise Manager (EM) to turn on caching, or export the `oim-config.xml` to make changes and then import it back to turn on caching.

Oracle recommends the following settings for the production environments for optimal and better performance. Using EM, go to `System Mbean` > `Application Defined Mbeans` > `oracle.iam` > `server:oim_server1` > `Application: oim` > `XMLConfig` > `Config` > `XMLConfig.CacheConfig` > `Cache` > `XMLConfig.CacheConfig.CacheCategoryConfig`, and do the following:

- Set the caching to `true` for all the components *except* the following two sections:

```
threadLocalCacheEnabled="false"
"StoredProcAPI" enabled="false"
```

- For non-clustered installation, set `clustered="false"`. For clustered installation, set `clustered="true"`.

> **Note:** Changing this value gets saved into the MDS database schema used by the Oracle Identity Manager servers. Therefore, change only once for multi-node/clustered installations.

### Enabling Cache Categories User_Org_Membership_And_Chain and ObjectDefinition

It is recommended that you enable the cache categories described in Instructions to Enable Cache Category Table 26–1, based on your Oracle Identity Manager version. Note that you do not need to enable these, if your Oracle Identity Manager version is not same as given in "**Applicable Release**" column in the following table:

***Table 26–1 Instructions to Enable Cache Category***

| Cache Category Name | Applicable Release | Instructions |
|---|---|---|
| User_Org_ Membership_ And_Chain | Oracle Identity Manager 11*g* Release 2 (11.1.2.1.0) | You can enable this cache category using Oracle Enterprise Manager (EM) or by editing the `oim-config.xml` configuration file. To do this, complete the following steps:<br>**Using EM**<br>1. Log in to EM.<br>2. Go to mbean **XMLConfig.CacheConfig** under **oracle.iam**, and set the value of attribute `Enabled` to `true`, if not already set to `true`. Mbean's Object name is "**oracle.iam:name=Cache,type=XMLConfig.CacheConfig,XMLConfig=Config,Application=oim,ApplicationVersion=11.1.2.0.0**".<br>3. Create a new cache category using mbean's **createCacheCategoryConfig** operation with the following parameters:<br>`enabled=true`<br>`expirationTime=3600`<br>`name=User_Org_Membership_And_Chain`<br>**Using oim-config.xml File**<br>1. Go to *$OIM_HOME*`/bin`.<br>2. Set the environment variable `OIM_ORACLE_HOME` appropriately.<br>3. Open the `weblogic.properties` file, and set the following properties in order to export the metadata file:<br>`wls_servername=`*`oim_server1`*<br>`application_name=OIMAppMetadata`<br>`metadata_to_loc=<`*`TMP_DIRECTORY`*`>`<br>`metadata_files=/db/oim-config.xml`<br>4. Run the following command script to export the `/db/oim-config.xml` metadata file:<br>`./weblogicExportMetadata.sh`<br>When prompted, enter the WebLogic credentials and the JNDI URL.<br>5. Open the *$TMP_DIRECTORY*`/db/oim-config.xml` file, and add the following in the `cacheCategoriesConfig` tag:<br>`<cacheCategoryConfig enabled="true"`<br>`expirationTime="14400"`<br>`name="User_Org_Membership_And_Chain"/>`<br>6. Open the `weblogic.properties` file, and set the following properties in order to import the modified metadata file:<br>`wls_servername=`*`oim_server1`*<br>`application_name=OIMAppMetadata`<br>`metadata_from_loc=<`*`TMP_DIRECTORY`*`>`<br>7. Run the following command to import the modified `/db/oim-config.xml` metadata file into MDS:<br>`./weblogicImportMetadata.sh`<br>When prompted, enter the WebLogic credentials and the JNDI URL. |

*Table 26–1 (Cont.) Instructions to Enable Cache Category*

| Cache Category Name | Applicable Release | Instructions |
|---|---|---|
| ObjectDefinition | Oracle Identity Manager 11*g* Release 2 (11.1.2.0.0) | You can enable this cache category using Oracle Enterprise Manager (EM). To do so, complete the following steps:<br>1. Log in to EM.<br>2. Go to mbean **XMLConfig.CacheConfig** under **oracle.iam**, and set the value of attribute `Enabled` to `true` for the cache category **ObjectDefinition**. |

**Note:** For more information on configuration change using Enterprise Manager, see "Using Enterprise Manager for Managing Oracle Identity Manager Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager* for information about how to make changes to this file.

#### 26.3.1.2 Purging the Cache

If you want to purge the cache, use the PurgeCache utility in the *OIM_HOME*/server/bin/ directory. This utility purges all elements in the cache.

**Note:**

- Purging is required when caching is enabled and if you make any system configuration changes. It is not required if caching is disabled.
- Before running the PurgeCache utility, navigate to the *OIM_HOME*/server/bin/ directory.

Before running the PurgeCache utility, you must run the *DOMAIN_HOME*/bin/setDomainEnv.sh script.

To use the PurgeCache utility, run `PurgeCache.bat` *CATEGORY_NAME* on Microsoft Windows or `PurgeCache.sh` *CATEGORY_NAME* on UNIX. The *CATEGORY_NAME* argument represents the name of the category that must be purged. For example, the following commands purge all FormDefinition entries from a system and its clusters:

```
PurgeCache.bat FormDefinition
PurgeCache.sh FormDefinition
```

To purge all Oracle Identity Manager categories, pass a value of "All" to the PurgeCache utility. It is recommended to clear all the categories.

**Note:** The wlfullclient.jar file must be in the classpath for the PurgeCache utility to run correctly.

### 26.3.2 Tuning the Application Server for Oracle Identity Manager

This section describes how to tune Oracle WebLogic Server for Oracle Identity Manager to improve performance. For additional Oracle WebLogic Server

performance tuning information, see *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

**Note:**

- All tuning parameter suggestions and values in this section are for reference purposes only. Values should be modified based on your requirement, application usage patterns, loads, and hardware specifications.
- Changing any of the settings may require you to restart the server.

- Tuning JVM Memory Settings for Oracle Identity Manager
- Tuning the JDBC Connection Pool for Oracle Identity Manager
- Tuning the Number of Message Driven Beans for Oracle Identity Manager
- Tuning the User Interface Threads for Oracle Identity Manager
- Disabling the Reloading of Adapters and Plug-in Configuration
- Changing the Number of Open File Descriptors for UNIX (Optional)
- Tuning the JVM Garbage Collection for Solaris Sparc T3 or T4

#### 26.3.2.1 Tuning JVM Memory Settings for Oracle Identity Manager

These settings should be used in addition to those described in Chapter 2.4, "Tuning Java Virtual Machines (JVMs)".

It is recommended to increase the heap and permgen memory for production environments as in Table 26–2 and monitor the memory usage pattern. Based on the usage, you can choose to increase or decrease the memory settings.

***Table 26–2 JVM Parameters to be set for Tuning JVM Memory Settings***

| JVM Parameter | HotSpot JVM | JRockit JVM |
|---|---|---|
| `Min. Heap Size (Xms)` | 4GB | 4GB |
| `Max Heap Size (Xmx)` | 4GB | 4GB |
| `PermSize (-XX:PermSize` | 500m | N/A |
| `PermGen size (-XX:MaxPermSize)` | 1GB | N/A |

To change the JVM memory setting:

1. If you have OIM version 11.1.2.1.0 or above, use `DOMAIN_HOME/bin/setOIMDomainEnv.sh` (Unix) or set `OIMDomainEnv.cmd` (Windows). If not, continue to use `DOMAIN_HOME/bin/setSOADomainEnv.sh` (Unix) or `setSOADomainEnv.cmd` (Windows) to change the heap size settings.
2. Change the value of `DEFAULT_MEM_ARGS` and `PORT_MEM_ARGS` from the default value and save.
3. Restart the OIM Server

**Note:** For a clustered or multi-node installation, repeat the above steps on all the install locations.

#### 26.3.2.2 Tuning the JDBC Connection Pool for Oracle Identity Manager

Oracle Identity Manager uses the oimOperationsDB and oimJMSStoreDS datasources deployed on Oracle WebLogic Server. By default, maximum connections is set at 50. You may have to increase this based on the requirement. To increase the capacity of the JDBC connection pools:

1. Open the WebLogic Server Administration Console.
2. For JDBC Datasource xlXADS:
   a. Click **Services**, **JDBC, Data Sources**, **oimOperationsDB**, and then click the **Connection Pool** tab.
   b. Adjust the Initial Capacity and Maximum Capacity based on requirement.
   c. Set the Inactive Connection Timeout parameter to 30.

   For JDBC Datasource xlDS:
   a. Click **Services**, **JDBC**, **Data Sources**, **oimJMSStoreDS**, and then click the **Connection Pool** tab.
   b. Adjust the Initial Capacity and Maximum Capacity based on requirement.
   c. Set the Inactive Connection Timeout parameter to 30.
3. Save and activate the changes.

> **Note:** Ensure that any increase in number of connections on the application server connection pools are compensated by database configuration changes. You might have to increase the MAX SESSIONS settings on Oracle Database.

#### 26.3.2.3 Tuning the Number of Message Driven Beans for Oracle Identity Manager

Oracle Identity Manager uses Message Driven Beans (MDBs) for processing all offline activities, such as reconciliation, auditing, requests, attestation, and for its internal kernel operations. By default, total of 80 MDB instances concurrently serve requests. However, based on the requirement, this can be increased by modifying the OIMMDBWorkManager configuration. To do so:

1. Login to WebLogic Administrative Console.
2. Navigate to **Environment**, **Work Managers**, and then to **MaxThreadsConstraint-1**.
3. Change the count from 80 to a higher number per your requirement.

#### 26.3.2.4 Tuning the User Interface Threads for Oracle Identity Manager

By default, Oracle Identity Manager provides 20 front-end thread configurations. These threads are used for serving front-end requests. To change the number of front-end thread configurations:

1. Login to WebLogic Administrative Console.
2. Navigate to **Environment**, **Work Managers**, and then to **MaxThreadsConstraint-0**.
3. Change the value of the count from 20 to number per your requirement.

#### 26.3.2.5 Disabling the Reloading of Adapters and Plug-in Configuration

By default, reloading of adapters and plug-in configuration are enabled for ease of development. These should be disabled in the production environment. To do so:

1. Export the /db/oim-config.xml file from MDS as described in "Exporting and Importing Configuration Files" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.
2. In the oim-config.xml file, replace the following:

```
<ADPClassLoaderConfig adapterReloadingEnabled="true" loadingStyle="ParentFirst"
reloadInterval="15" reloadingEnabled="true">
```

   With:

```
<ADPClassLoaderConfig adapterReloadingEnabled="false"
loadingStyle="ParentFirst" reloadInterval="15" reloadingEnabled="false">
```

3. Replace the following:

```
<storeConfig reloadingEnabled="true" reloadingInterval="20"/>
```

   With:

```
<storeConfig reloadingEnabled="false" reloadingInterval="20"/>
```

4. Save the oim-config.xml file and import it back to MDS.

#### 26.3.2.6 Changing the Number of Open File Descriptors for UNIX (Optional)

WebLogic limits the number of open file descriptors in the *WEBLOGIC_HOME*/common/bin/commEnv.sh script to 1024. In some cases, if there is a large number of concurrent users, WebLogic may throw the "TOO MANY OPEN FILES" exception. If you receive this error, then consider increasing the limit beyond 1024 in the script. Ensure that the operating system is able to handle the increase in the number of open files.

#### 26.3.2.7 Tuning the JVM Garbage Collection for Solaris Sparc T3 or T4

To tune the JVM garbage collection for Solaris Sparc T3 or T4:

1. In a text editor, open the setSOADomainEnv.sh or setSOADomainEnv.cmd file in the *DOMAIN_HOME*/bin/ directory.
2. Set the value of USER_MEM_ARGS similar to the following:

> **Note:** The values shown for USER_MEM_ARGS are examples. You can change the values based on your requirement.

```
USER_MEM_ARGS="-Xms3048m -Xmx3048m -Xmn1648m -Xss256k -XX:PermSize=384m
-XX:MaxPermSize=384m"
```

3. Set the value of JAVA_OPTIONS similar to the following:

> **Note:** The values shown for JAVA_OPTIONS are examples. You can change the values based on your requirement.

```
JAVA_OPTIONS="-Xnoclassgc -XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=90
 -XX:PermSize=350m -XX:MaxPermSize=350m -XX:+AggressiveOpts
 -XX:+UseParallelOldGC -XX:ParallelGCThreads=8 -XX:+PrintGCDetails
 -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps  -XX:ReservedCodeCacheSize=64m
 -XX:CICompilerCount=8 -XX:+AlwaysPreTouch -XX:+PrintReferenceGC
 -XX:+ParallelRefProcEnabled -XX:-UseAdaptiveSizePolicy
```

```
-XX:+PrintAdaptiveSizePolicy -XX:+DisableExplicitGC"
```

4. Save and close the file.

## 26.3.3 Tuning Database Parameters for Oracle Identity Manager

This section describes one sample configuration and outlines the principles for tuning Oracle Database for Oracle Identity Manager. For general database tuning information, see Tuning Database Parameters.

Oracle Identity Manager has many configuration options. The best way to identify bottlenecks and optimize performance is to monitor key database performance indicators in your production environment and adjust the configuration accordingly. Review the monitoring tasks described in Monitoring Oracle Identity Manager Performance and then use the guidelines in this section to help you choose the initial baseline database configuration.

> **Note:** It is important that you maintain the baseline database tuning parameters when working with Oracle Identity Manager. See the *Oracle Database Performance Tuning Guide 11g Release 1 (11.1)* for information on setting Oracle Database instance parameters.

### 26.3.3.1 Sample Instance Configuration Parameters

Table 26–3 provides information on some important performance-related database initialization parameters.

SGA,PGA size are limited by the underlying operating system restrictions on the maximum available memory in some platforms. See Support Note: Oracle Database Server and the Operating System Memory Limitations [ID 269495.1].

> **Note:** For the Database Instance Parameters listed in Table 26–3, any one of the following memory management approaches can be used based on the Oracle Database versions:
>
> - Using Automatic Memory Management feature available in Oracle Database 11*g*: Here, the MEMORY_TARGET and MEMORY_MAX_TARGET parameters can be used to manage the SGA and PGA together.
> - Using Automatic Shared Memory Management (ASMM) available in Oracle Database 10*g* onward: Here, the SGA components can be managed by specifying the SGA_TARGET and SGA_MAX_SIZE parameters. PGA is managed separately through PGA_AGGREGATE_TARGET.
>
> You should set the processes parameter to accommodate the following connection pool requirements and few extra connections for external programs:
>
> - Connection pool size of XA data-source configured in Application Server
> - Connection pool size for non-XA data-source configured in Application Server
> - Direct database connection pool size configured in xlconfig.xml

*Table 26–3 Sample Configuration Parameters*

| Parameter | Recommended Initial Settings for Oracle Database 11*g* |
|---|---|
| memory_target | Using Automatic Memory Management feature in Oracle Database 11*g*, the MEMORY_TARGET and MEMORY_MAX_TARGET parameters can be used to manage the SGA and PGA together.<br>Following are the memory settings for all the releases of IDM:<br>SGA_target - 4G<br>PGA_AGGREGATE_TARGET - 2G<br>You can unset the MEMORY_TARGET and MEMORY_MAX_TARGET from 11g onwards.<br>When considering MEMORY_TARGET for managing the database memory components, SGA_TARGET and PGA_AGGREGATE_TARGET can be left unallocated, which is 0. |
| db_keep_cache_size | 800M |
| cursor_sharing | FORCE |
| open_cursors | 800 |
| session_cached_cursors | 800 |
| query_rewrite_integrity | TRUSTED |
| query_rewrite_enabled | TRUE |
| processes | Based on connection pool settings |
| MAX_DISPATCHERS | 0 |
| MAX_SHARED_SERVERS | 0 |
| DISK_ASYNCH_IO | True |

#### 26.3.3.2 Physical Data Placement

The basic installation of Oracle Identity Manager uses three physical tablespaces to store the OIM database objects:

- Data Tablespace to store the data of tables, their indexes and other objects.
- LOB Tablespace to store OIM Orchestration LOB data.
- Archival Tablespace to store OOTB Archival Tables of the OIM Entities catering to the Real-time Purge feature.

**Tip:** To minimize disk space consumption, Oracle recommends the following:

During the initial startup phase of the deployment, Oracle Identity Manager tablespace is expected to grow at the rate 20G for every hundred thousand users reconciled into Oracle Identity Manager. LOB tablespace grows at around 30% of the size of main Oracle Identity Manager tablespace for the same users. Depending on the usage of orchestration in Oracle Identity Manager, which affects the LOB tablespace growth, the LOB tablespace can grow at a rate of 60% to 100% of the main tablespace in scenarios where orchestration is widely used.

Database administrators must monitor the exact growth rate in the real system for efficient disk space management.

For better performance, create multiple locally managed tablespaces and store each category of database object in a dedicated tablespace. This storage optimization helps efficient data access. The tables that are frequently accessed and have potential growth are highlighted in the following sections. Oracle recommends that you place these tables in their own dedicated tablespace(s).

Note that the tables highlighted in the following sections generally grow bigger and are accessed frequently in a typical Oracle Identity Manager deployment. In addition, you can use performance metrics to identify tables that are accessed frequently (hot tables). To reduce I/O contention, move hot tables to dedicated tablespaces.

**Note:** Oracle Identity Manager offers archival and purge solution in both Real-time online mode and Command Line mode to contain the data growth in most of these tables. See "Using the Archival Utilities" in *Using the Archival and Purge Utilities for Controlling Data Growth* for more information.

**26.3.3.2.1 Tasks Tables** Oracle Identity Manager stores provisioning and approval task details in the following tables. These tables have lot of potential to grow big overtime. It is recommended to group these in one or more dedicated tablespaces.

- OSI
- OSH
- SCH

**26.3.3.2.2 Reconcliation Tables** The reconciliation schema of Oracle Identity Manager has both static and dynamic tables. The following is a list of static tables. The dynamic tables can be identified by querying the RECON_TABLE_NAME column in the RECON_TABLES table.

- RECON_ACCOUNT_OLDSTATE
- RECON_BATCHES
- RECON_CHILD_MATCH
- RECON_EVENTS
- RECON_EVENT_ASSIGNMENT

- RECON_EXCEPTIONS
- RECON_HISTORY
- RECON_JOBS
- RECON_TABLES
- RECON_UGP_OLDSTATE
- RECON_USER_OLDSTATE
- RECON_ACCOUNT_MATCH
- RECON_ORG_MATCH
- RECON_ROLE_HIERARCHY_MATCH
- RECON_ROLE_MATCH
- RECON_ROLE_MEMBER_MATCH
- RECON_USER_MATCH
- RA_LDAPUSER
- RA_MLS_LDAPUSER
- RA_LDAPROLE
- RA_MLS_LDAPROLE
- RA_LDAPROLEMEMBERSHIP
- RA_LDAPROLEHIERARCHY

If your environment generates a large amount of reconciliation data, then move these tables to one or more dedicated tablespace(s).

**26.3.3.2.3 Audit Tables** Oracle Identity Manager audits the transactions based on the audit level setting. Most of the audit levels are likely to increase data growth significantly. Oracle recommends storing audit tables in their own tablespace. Oracle Identity Manager audit tables are of two categories. Following are the tables that store audit data in XML format. In this list, UPA table is especially expected to grow big and it is important to place it in a dedicated tablespace.

- UPA
- GPA

The user profile audit data is stored in the following flat structured tables. These tables are used by Oracle Identity Manager historical reports for compliance reporting. It is recommended to store these tables and their indexes in a dedicated tablespace.

- UPA_FIELDS
- UPA_GRP_MEMBERSHIP
- UPA_RESOURCE
- UPA_USR
- UPA_UD_FORMS
- UPA_UD_FORMFIELDS

You can use the Archival Utilities to maintain a large growth table. For more information, see *Using the Archival Utilities* in Oracle Fusion Middleware System Administrator's Guide for Oracle Identity Manager.

**26.3.3.2.4 Redo-Log Files** Depending on the reconciliation processes configured in Oracle Identity Manager, the volume of database transactions and commits during a reconciliation run can be high. Oracle recommends that you use multiple redo-log files. The total allocated redo-log space should be 1 GB to 2 GB.

Oracle recommends use of at least three redo log groups with redo log members with minimum size of 500 MB for each. The multiplexing and the exact number of members and disk space for each member can be considered in accordance with the planning for failure.

**26.3.3.2.5 Keep Pool Changes** By default, Oracle Identity Manager assigns frequently referenced small tables to be cached in the database by using a keep pool buffer. See db_keep_cache_size in Table 26–3. If your installation contains more than 50,000 users, then Oracle recommends that you use the default database buffer for USR and PCQ tables instead of the keep pool buffer. You can use the following commands to put these tables in default buffer pool.

```
ALTER TABLE USR STORAGE(buffer_pool default);
ALTER TABLE PCQ STORAGE(buffer_pool default);
```

## 26.3.4 Tuning Oracle Internet Directory

To ensure that the Oracle Identity Manager is performing at the optimal level, it is important to tune the Oracle Internet Directory as described in Chapter 23, "Oracle Internet Directory Performance Tuning".

## 26.3.5 Tuning Application Module (AM) for User Interface

Application Module tuning is a critical setting which affects the UI performance. You must ensure that the recommended application module settings for Oracle Identity Manager are set in the `setDomainEnv.sh` file. These settings are already set out-of-box (OOB) in later releases of Oracle Identity Manager 11g Release 2 (11.1.2). To add the recommended application module settings for Oracle Identity Manager, do the following:

1. Open the file *$DOMAIN_HOME*`/bin/setDomainEnv.sh` in a text editor.
2. In the `setDomainEnv.sh` file, find the following lines:

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

3. Change the first line to 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:** These recommended settings assume 100 concurrent users per node. If your number of concurrent users is different, use the following formula to change `Djbo.ampool.maxavailablesize`:

`Djbo.ampool.maxavailablesize = # of concurrent users + 20%`

---

4. Save the `setDomainEnv.sh` file.
5. Restart the WebLogic Administration Server and the Oracle Identity Manager Managed Servers.

For more information on AM Pool tunings, see section 8.3.5 "Application Module Pooling" in the *Oracle Fusion Middleware Performance and Tuning Guide*.

## 26.4 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 Oracle Identity Manager performance.

- Reconciliation Tuning
- Tuning LDAP Synchronization

### 26.4.1 Reconciliation Tuning

Three distinct process stages or functional modules come into play during the end-to-end reconciliation flow. The following are the three functional modules or stages that need to be optimized separately, but in relation to each other, to achieve complete performance optimization:

- The Target System And The Connector

  The Connector fetches data from the target system, and invokes reconciliation create event APIs to create events and event data in reconciliation staging tables in the OIM database schema.

- OIM Reconciliation Engine

  The OIM reconciliation engine extracts data from the staging tables and reconciles into OIM. The process includes verification, matching of data, and taking actions based on the rules. The engine uses database's bulk collection mechanism to do all of the above processing in bulk.

- Oracle Identity Manager Post-processing for Reconciliation

  Post-processing stage kicks in after reconciliation engine has completed processing of incoming data from the target. During this stage, OIM kernel orchestrations get triggered to execute event-handlers to do things like default password generation as per policy, role assignment, resource provisioning, audit processing and so on.

This section includes the following topics:

- Target System And Connector Tuning
- Database Indexes For Recon Matching Rules
- Oracle Identity Manager Post-processing for Reconciliation

#### 26.4.1.1 Target System And Connector Tuning

This section describes the tuning that needs to be applied on your target systems as well as Oracle Identity Manager Connectors.

**Oracle Internet Directory**

During a reconciliation run, all changes in the target system records are reconciled into Oracle Identity Manager. Depending on the number of records to be reconciled, this process may require a large amount of time. In addition, if the connection breaks

during reconciliation, then the process would take longer to complete. It is recommended that "paged reconciliation" is configured to optimize performance.

To configure paged reconciliation, you must specify a value for the `PageSize` attribute of the user reconciliation scheduled task. The default value of `100` for `PageSize` suits for most of the scenarios.

> **Note:** OID LDAP Server (the target system in this case) v10.1.4 or later versions support the paged reconciliation related LDAP operations.

**SAP**

It is recommended that you use a reconciliation batch size of `100`.

**Active Directory (11.1.1.5.0 and 11.1.1.6.0 Connector)**

- Performance improvement patch
  - If you are using Active Directory 11.1.1.5.0, make sure that you apply patch # 15916848. You can download the patch from My Oracle Support. For patching instructions, refer to the Readme that is available with the patch.
  - If you are using Active Directory 11.1.1.6.0, download the patch # 15916848 from My Oracle Support. Import only the `ReconAttributeMap.xml` that is provided as part of the patch, using the deployment manager. You can ignore `ActiveDirectory.Connector.dll` provided in the patch, as it is updated in the 11.1.1.6.0 version itself. For patching instructions, refer to the Readme that is available with the patch.
- Configuring the reconciliation engine to skip the ignore event API

  The default behavior would be to first check to create a recon event or to ignore it for each of the user records returned by the connector. This process involves comparing the values of all the attributes of the user coming in from the connector against the values stored in the OIM database. To ignore this, open the lookup definition `Lookup.Configuration.ActiveDirectory` and add below entry.
  - `Code Key: Ignore Event Disabled`
  - `Decode: true`

  > **Note:** You must evaluate the pros and cons of disabling the ignore event API call before you make the above changes.

- Batching

  If batching is used in the AD connector, then the result set needs to be sorted. Therefore, batching can be used when number of records to be reconciled is less than `10000`. The recommended batch size is `500`.
- Paging
  - When number of records to be reconciled is more than `10000`, use the `Page Size Configuration` property present in `Lookup.Configuration.ActiveDirectory` and `Lookup.Configuration.ActiveDirectory.Trusted`.

  - If paging is configured to be used, then you must make sure that no value is specified for the scheduled task parameters - `Batch Size`, `Batch Start`, `Number of Batches`, `Sort By`, and `Sort Direction`.
  - Paging splits the entire result set of a query into smaller subsets called, appropriately enough, pages. In general, it is recommended to set this value to the maximum page size for simple searches. By setting the page size to the maximum value, you can minimize the network round trips necessary to retrieve each page, which tends to be more expensive operation for simple searches. If you specify a `PageSize` greater than the `MaxPageSize` of the target system, the Active Directory server ignores it and uses the `MaxPageSize` instead. No exception is generated in this case. In some cases, you might need to specify a smaller page size to avoid timeouts or overtaxing the server. Some queries are especially expensive. Therefore, limiting the number of results in a single page can help avoid this. For the Active Directory Connector, use the default value `1000` for the best performance.

- Filters

  It is recommended to use `Filters` and provide the value for the `Search Base`, if a specific set of records is to be retrieved from the target. Filter provided in the scheduled task is converted into LDAP query. The filters help narrow down the search, making the searching and processing of the data quicker. For more information about the filters, refer to the Active Directory Connector Documentation.

- For the reconciliation in the forest topology, you can use connector for reconciling the data from the complete forest (via Global Catalog Server) or you can use the connector for reconciling the data from the specific domain or domain controller. It is recommended to use the second approach whenever the data from the specific data center is to be reconciled, instead of using first option with search base.

  For example:

  Assume that there are 10 data centers in the Active Directory forest namely DC1, DC2, … , DC10. To reconcile data from an organization (tempOrg) which is present on DC2, you have use one of the following approaches:

  **a.** Use Global Catalog and provide the DN of the organization in the Search Base.

  **b.** Use DC2 and provide the DN of the organization in the Search Base.

  It is recommended to use the second approach for better performance.

#### 26.4.1.2 Database Indexes For Recon Matching Rules

Reconciliation uses matching algorithm to find if the user/account/role/organization for which the change is requested, already exists in OIM. The matching algorithm compares the data in set of columns in OIM with the data in target staging table columns. The columns that contain the matching rules are defined in the reconciliation profile and they are defined at run-time. To improve the performance of the matching operation, there must be correct indexes created on the matching rule columns.

To illustrate the recommended method of identifying the appropriate indexes, a sample Active Directory (AD) user profile present in the Meta Data Store (MDS) repository is taken as an example. This example covers the following:

- Selecting Indexes For Trusted Source Reconciliation
- Selecting Indexes For Target Source Reconciliation
- Selecting Indexes For Target Source Reconciliation With Multi-Valued Data

**Note:** Starting OIM 11*g* Release 2 (11.1.2.1.0), the indexes are automatically created in some cases where possible. It is still recommended to follow the below procedure and make sure that all of the indexes required for reconciliation matching rule are in place.

### Selecting Indexes For Trusted Source Reconciliation

To select indexes based on the matching rule criteria in trusted source reconciliation, you must complete the following steps:

1. Open the Active Directory user profile file in a text editor. You can open Active Directory user profile using `Validate Recon Profile` test present in the diagnostic dashboard, or by using `Validate Recon Profile` MBean present in EM.
2. Search for `ownerMatchingRuleWhereClause` or matchingRule for all entities:

   ```
   ownerMatchingRuleWhereClause = (((UPPER(USR.USR_LOGIN)=UPPER(RA_
   ADUSER7.RECON_USERID5A729570)) OR (UPPER(USR.USR_UDF_OBGUID)=UPPER(RA_
   ADUSER7.RECON_OBJECTGUID))))
   ```

3. After identifying the columns constituting the matching rule in the profile, create the indexes accordingly.

   For example, following indexes are needed for matching rule in the above example.

***Table 26–4 Table Names and Columns to be Indexed***

| Table Name | Column to be Indexed |
|---|---|
| USR | UPPER(USR_LOGIN) |
| USR | UPPER(USR.USR_UDF_OBGUID) |
| RA_ADUSER7 | UPPER(RECON_USERID5A729570) |
| RA_ADUSER7 | UPPER(RA_ADUSER7.RECON_OBJECTGUID) |

**Note:**

- It is important that the indexes are created along with functions like `UPPER`, `SUBSTR` in the matching rule. In Table 26–4, `UPPER` is the function used on all columns.
- Some of the columns and functions might have been indexed already. In Table 26–4, `USR` table should already have function-based index on `UPPER(USR_LOGIN)`.

### Selecting Indexes For Target Source Reconciliation

To select indexes based on the matching rule criteria in target resource reconciliation, you must complete the following steps:

1. Open the Active Directory user profile file in a text editor. You can open Active Directory user profile using `Validate Recon Profile` test present in the diagnostic dashboard, or by using `Validate Recon profile` MBean present in EM.
2. Search for account search tag `<matchingruleWhereClause>`:

```
<matchingruleWhereClause>((UD_ADUSER.UD_ADUSER_OBJECTGUID=RA_
ADUSER7.RECON_OBJECTGUID))</matchingruleWhereClause>
```

3. After identifying the columns constituting the matching rule in the profile, create the indexes accordingly.

   For example, following indexes are needed for matching rule in the above example.

***Table 26–5 Table Names and Columns to be Indexed***

| Table Name | Column to be Indexed |
|---|---|
| UD_ADUSER | UD_ADUSER_OBJECTGUID |
| RA_ADUSER7 | RECON_OBJECTGUID |

**Note:**

- It is important that the indexes are created along with functions like `UPPER`, `SUBSTR` in the matching rule.
- Some of the columns and functions might have been indexed already.

### Selecting Indexes For Target Source Reconciliation With Multi-Valued Data

To select indexes based on the matching rule criteria in target resource reconciliation with multi-valued data, you must complete the following steps:

1. Open the Active Directory user profile file in a text editor. You can open Active Directory user profile using `Validate Recon Profile` test present in the diagnostic dashboard, or by using `Validate Recon profile` MBean present in EM.
2. For entitlements, search for the `<matchingruleWhereClause>` tag under `<childreconeventdata>`:

   ```
   <matchingruleWhereClause>((UD_ADUSRC.UD_ADUSRC_GROUPNAME=RA_UD_
   ADUSRC.RECON_MEMBEROF))</matchingruleWhereClause>
   ```

3. After identifying the columns constituting the matching rule in the profile, create the indexes accordingly. For example, following indexes are needed for matching rule in the above example.

***Table 26–6 Table Names and Columns to be Indexed***

| Table Name | Column to be Indexed |
|---|---|
| UD_ADUSRC | UD_ADUSRC_GROUPNAME |
| RA_UD_ADUSRC | RECON_MEMBEROF |

**Note:**

- It is important that the indexes are created along with functions like `UPPER`, `SUBSTR` in the matching rule.
- Some of the columns and functions might have been indexed already.

#### 26.4.1.3 Oracle Identity Manager Post-processing for Reconciliation

Table 26–7 lists some of the important out-of-the-box event handlers that are invoked during post-processing of reconciliation.

***Table 26–7 Event Handlers and Their Descriptions***

| Event Handler | Description |
|---|---|
| `AccountReconAuditHandler` | Responsible for Auditing account/target reconciliation changes |
| `ReconScheduledTaskAccountHandler` | Trigger workflows associated with account/target reconciliation |
| `ReconScheduledTaskUserHandler` | Trigger workflows associated with trusted reconciliation |
| `ReconUserDisplayNameHandler` | Generates custom display name for trusted reconciliation |
| `ReconUserLoginHandler` | Generates custom login during for reconciliation |
| `ReconUserPasswordHandler` | Generates custom passwords for trusted reconciliation |
| `UserCreateLdapPostProcessHandler` | Creates user in LDAP if LDAP synchronization is enabled |
| `UserUpdateLdapPostProcessHandler` | Updates user in LDAP if LDAP synchronization is enabled |

You can find the rest of out-of-the-box and custom event handlers in DMS metric page of WebLogic Application Server. Use the following URL to go to the DMS metric page:

`http://servername:port/dms`

In this URL, `port` refers to the WebLogic Administration Server port. To log in, you must use the WebLogic admin credentials.

After you log into the DMS metric page, click on **OIM_EventHandler** to see the list of event handlers and their processing time metrics. You can use these metrics to identify event handlers that may need to be optimized.

### 26.4.2 Tuning LDAP Synchronization

Tuning performance in Oracle Identity Manager involves the following:

- Increasing the Max Connection Pool for Oracle Identity Manager
- Increasing the LDAP Synchronization Batch Size

#### 26.4.2.1 Increasing the Max Connection Pool for Oracle Identity Manager

To increase the max connection pool for Oracle Identity Manager:

1. Login to Oracle Identity System Administration.
2. On the left pane, under Configuration, click **IT Resource**. The Manage IT Resource page is displayed in a new window.
3. From the IT Resource Type list, select **Directory Server**, and then click **Search**.
4. For the Directory Server IT resource, click **Edit**. The Edit IT Resource Details and Parameters page is displayed.
5. Change the value of the following configuration parameters to 500:

- Initial pool size: 500
- Minimum pool size: 500
- Maximum pool size: 500

6. Click **Update**.

**26.4.2.1.1 Increasing the LDAP Synchronization Batch Size** To increase the LDAP synchronization batch size, set the batch size of the following LDAP synchronization reconciliation scheduled jobs to 1000:

- LDAP User Create and Update Reconciliation
- LDAP Role Create and Update Reconciliation
- LDAP Role Hierarchy Reconciliation
- LDAP Role Membership Reconciliation

> **Note:** For details about the LDAP scheduled jobs, see "LDAP Scheduled Tasks" in *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

**26.4.2.1.2 Setting Configuration Parameters in OVD** When LDAP synchronization with OVD configured for OID is enabled in Oracle Identity Manager, the configuration parameters in OVD, as listed in Table 26–8, must be set:

***Table 26–8 Configuration Parameters in OVD***

| Name | Parameter | Value |
|---|---|---|
| OVD general | `Listeners - LDAP Endpoint` | 50 |
| | `Listeners - LDAP SSL Endpoint` | 50 |
| User Adapter | `Max Pool Size` | 500 |
| | `Operation Timeout` | 1500000 |
| | `Max Pool Wait` | 1000 |
| Changelog adapter | `Max Pool Size` | 500 |
| | Operation Timeout | 1500000 |

**26.4.2.1.3 Setting Configuration Parameters in OID** When LDAP synchronization with OVD/OID is enabled in Oracle Identity Manager, the configuration parameters in OID, as listed in Table 26–9, must be set:

***Table 26–9 Configuration Parameters in OID***

| Name | Parameter | Value |
|---|---|---|
| Max Number of DB Connections | `orclmaxcc` | 10 |
| Number of Processes | `orclserverprocs` | 2 - 4 |
| Skip Referral Process | `orclskiprefinsql` | 1 |
| LDAP Connection Timeout | `orcllldapconntimeout` | 60 |
| Enable MatchDN Processing | `orclmatchdnenabled` | 0 |
| Enable Entry Cache | `orclcacheenabled` | 0 |

To modify the attributes in Table 26–9, use the following syntax:

```
ldapmodify -h HOST_NAME -p PORT_NUMBER -D cn=orcladmin -w PASSWORD -v <<EOF
dn: cn=oid1,cn=osdldapd,cn=subconfigsubentry
```

**26.4.2.1.4 Setting Configuration Parameters in Identity Virtualization Library (libOVD)** When LDAP synchronization with Identity Virtualization Library (libOVD) configured for OID is enabled in Oracle Identity Manager, the configuration parameters in Identity Virtualization Library (libOVD), as listed in Table 26–10, must be set:

> **Note:** You can manage the Identity Virtualization Library (libOVD) tuning parameter configuration by using the WLST command.

***Table 26–10 Configuration Parameters in Identity Virtualization Library (libOVD)***

| Name | Parameter | Value |
|---|---|---|
| User Adapter | `Max Pool Size` | 500 |
| | `Operation Timeout` | 1500000 |
| | `Max Pool Wait` | 1000 |
| Changelog adapter | `Max Pool Size` | 500 |
| | `Operation Timeout` | 1500000 |

> **See Also:** "Enabling Access Logging in Identity Virtualization Library (libOVD)" in the *Oracle Fusion Middleware Integration Guide for Oracle Identity Management* for information about enabling access logging in Identity Virtualization Library (libOVD) to capture all requests and responses flowing through Identity Virtualization Library (libOVD), which can be very useful in triaging performance issues.

**26.4.2.1.5 Setting Configuration Parameters in WebLogic Server and JDBC** For information about setting configuration parameters in Oracle WebLogic Server and JDBC, see Section 26.3.2, "Tuning the Application Server for Oracle Identity Manager".

# 27

# Oracle Adaptive Access Manager Performance Tuning

This chapter provides guidelines for tuning and sizing Oracle Adaptive Access Manager (OAAM). It covers these topics:

- Section 27.1, "About Oracle Adaptive Access Manager"
- Section 27.2, "Performance Considerations"
- Section 27.3, "Monitoring Oracle Adaptive Access Manager"
- Section 27.4, "Basic Tuning Considerations"
- Section 27.5, "Advanced Tuning Considerations"
- Section 27.6, "Specific Use Cases That Require Additional Tuning"

> **Note:** Before tuning Oracle Adaptive Access Manager review and implement the general tuning configurations discussed in Chapter 2, "Top Performance Areas".

## 27.1 About Oracle Adaptive Access Manager

Oracle Adaptive Access Manager (OAAM) provides real-time or batch risk analysis and adaptive authentication capabilities to actively prevent fraud. Out of the box integrations with Oracle Identity Manager (OIM) and Oracle Access Manager (OAM) secure web single sign-on and self-service password management flows with adaptive authentication.

For more information on Oracle Adaptive Access Manager, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

## 27.2 Performance Considerations

Effective Oracle Adaptive Access Manager performance tuning starts with a good understanding of its usage and general performance issues. Before you begin tuning Oracle Adaptive Access Manager, review this section as well as the recommendations discussed in Top Performance Areas:

*Table 27–1 Oracle Adaptive Access Manager Performance Considerations*

| | |
|---|---|
| Number of users | Understanding the overall user population size; group, membership and attribute counts; data types, and configuration parameters of the LDAP and database is essential for properly tuning Oracle Adaptive Access Manager. See Performance Planning for more information on using population data to improve performance. |
| Daily activity usage | It is important to know how many users are active during a 24-hour period and the expected traffic. Spikes in usage may require additional tuning to avoid performance issues. See Monitoring Oracle Fusion Middleware for more information on collecting performance data. |
| Hardware resources and topology | Like any application deployed in demanding, real-time environments, proper server sizing and configuration is critical for acceptable Oracle Adaptive Access Manager performance. Ensuring that your hardware is sufficient to prevent bottlenecks is a key factor in performance tuning Oracle Adaptive Access Manager. See Securing Sufficient Hardware Resources for more information on optimizing hardware resources. |
| Protected applications | Knowing which applications Oracle Adaptive Access Manager is protecting and how that protection is modeled is an important consideration when tuning. Specifically you should understand how the applications are being protected: using WebGates (10g,11g, or 11gPS1); mod_osso; custom AccessGates; or a combination. |
| JVM and garbage collection | Optimal performance of Oracle Adaptive Access Manager depends on correctly tuning JVM heap sizes and garbage collection. See Configuring Garbage Collection and Specifying Heap Size Values in Tuning Java Virtual Machines (JVMs)for more information. |

## 27.3 Monitoring Oracle Adaptive Access Manager

To identify performance bottlenecks, you should monitor real-time performance metrics for Oracle Adaptive Access Manager. The following sections describe ways to monitor Oracle Adaptive Access Manager and what you should be monitoring to ensure Oracle Adaptive Access Manager is meeting your performance requirements:

### 27.3.1 Enabling Dynamic Monitoring Service (DMS)

Consider enabling Dynamic Monitoring Service (DMS) performance instrumentation which can tell you the latency and throughput of functional and operational metrics. DMS can identify components that are either processing a heavier load or taking longer than usual to service requests. See Viewing DMS Metrics for more information on determining the overall time to process calls to various components

### 27.3.2 Using the Oracle Adaptive Access Manager Admin Console Dashboard

Oracle Adaptive Access Manager Admin Console provides a performance dashboard that shows the performance of the traffic that is entering the system. A trending graph

is shown of the different types of data based on performance. For more information on accessing and using the Oracle Adaptive Access Manager Admin Console, see "OAAM Admin Console and Controls" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

Specifically you should use the dashboard to monitor performance of the following:

- Average execution time of Checkpoints (first) and Rules (second) and their trending data.
- Response time of APIs and their trending data
- Statistics such as the number of logins and transactions can tell you when the system is handling larger numbers of users. This information will help you determine your overall performance strategy.

### 27.3.3 Monitoring Oracle Adaptive Access Manager Server Logs

Consider monitoring the Oracle Adaptive Access Manager Server logs to check for messages related to slow-running SQL statements and errors. For example, If you notice numerous messages that are related to high response times of the SQL statements, then that usually means there are issues in the database. In this case the DBA should look at the database and investigate further to narrow down the issue.

To view the server log files, you can use Oracle Enterprise Manager Fusion Middleware Control or go to the Oracle Adaptive Access Manager home directory and look at the Oracle Adaptive Access Manager Server log files to find these issues.

For more information on using Oracle Adaptive Access Manager Server logs, see "Configuring Logging Output" in *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

### 27.3.4 Analyzing Automatic Workload Repository (AWR) Reports

To monitor the health of Oracle Adaptive Access Manager database the AWR reports can be used to pinpoint issues in the database. You can use Oracle Enterprise Manager to view these reports.

## 27.4 Basic Tuning Considerations

This section provides the basic tuning considerations for Oracle Adaptive Access Manager. It contains the following tuning recommendations:

- Using Purge Scripts to Improve Performance
- Tuning Database Parameters for OAAM
- Tuning Oracle Internet Directory
- Tuning Applications

### 27.4.1 Using Purge Scripts to Improve Performance

OAAM uses purge scripts to archive and purge different sets of transactional tables in the OAAM database. Purging releases disk space in the database for current data and deletes obsolete data. The purge process can be based on the age of the data or the type of data. By default OAAM purge scripts will archive data that will be deleted during the purge process. Targeted transaction and entity data archive and purge using the OAAM Admin Console is available in OAAM 11.1.2.

## 27.4.2 Tuning Database Parameters for OAAM

When tuning Oracle Adaptive Access Manager, you must first review the baseline tuning parameters for your database. For more information, see Section 2.6, "Tuning Database Parameters".

## 27.4.3 Tuning Oracle Internet Directory

To ensure that the Oracle Adaptive Access Manager is performing at the optimal level, it is important to tune the Oracle Internet Directory as described in Chapter 23, "Oracle Internet Directory Performance Tuning".

## 27.4.4 Tuning Applications

Oracle Adaptive Access Manager performs best when the following application parameters are set:

- Tuning Java Virtual Machine Parameters
- Tuning JDBC Connection Pool for OAAM
- Setting Logging Levels

### 27.4.4.1 Tuning Java Virtual Machine Parameters

To optimize Oracle Adaptive Access Manager, See the Java Virtual Machine tuning recommendations discussed in Section 2.4, "Tuning Java Virtual Machines (JVMs)".

### 27.4.4.2 Tuning JDBC Connection Pool for OAAM

JDBC Datasource `OAAM_ADMIN_DS` is used by the Oracle Adaptive Access Manager administration server for configuration. JDBC Datasource `OAAM_SERVER_DS` is used by the Oracle Adaptive Access Manager administration server for customer authentication and other transactional purposes. To increase the capacity of the JDBC connection pools consider the following:

| Parameter | Description |
|---|---|
| `OAAM_ADMIN_DS` | Set the Initial Capacity to one half of the Maximum Capacity. For example, consider setting the Initial Capacity to 50 and Maximum Capacity to 100. |
| `OAAM_SERVER_DS` | Set the same value for Initial Capacity and Maximum Capacity. For example, consider setting the Initial Capacity and Maximum Capacity to 100. |

For more information on using the Oracle WebLogic Administration Console to modify connection pools, see "Configure JDBC generic data sources" in the Oracle WebLogic Console online help.

### 27.4.4.3 Setting Logging Levels

For Oracle Adaptive Access Manager performance testing and production environments, consider using the lowest acceptable logging level whenever possible.

By default, log messages are written to the access.log file only when logging is set to NOTIFICATION:1. To maintain performance, consider keeping the default log level ERROR:1 (SEVERE) or use WARNING:1 (WARNING) to limit the amount of information written to the access.log file.

For more information on tuning the Oracle WebLogic Server log files, see the *Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server*.

## 27.5 Advanced Tuning Considerations

The following Oracle Adaptive Access Manager tuning considerations are provided as a guide. Always consult your own use case scenarios to determine if these configurations should be used in your deployment.

### 27.5.1 Disabling Tracker Node History Logging

If the history of the device is not required, then device history logging can be turned OFF by setting the property `bharosa.trackernodehistory.enable` to `false` using Oracle Adaptive Access Manager Admin Console.

### 27.5.2 Tuning Rule Logging Entry Creation

By default detailed rule logging takes place whenever a rule takes more than 2000 milliseconds (2 seconds) to execute. This is controlled through the `vcrypt.tracker.rulelog.detailed.minMillis` property.

### 27.5.3 Tuning Auto-learning Data Collection

The Auto-learning feature tracks transactions and authentications being performed by different actors based on patterns you create. This process establishes what is "normal" or average behavior for an individual or a population. By default, Auto-learning collects data for hourly, daily granularity that is not used by the out-of-the-box patterns. If there are no custom patterns that use hourly, daily granular data, then that data collection can be disabled by setting the following properties to **false**:

```
tracker.wf.createHourlyEntries
tracker.wf.createDailyEntries
```

> **Note:** When auto-learning is disabled, no pattern-based risk analysis will be performed. Consider this before you disable auto-learning as the risk analysis may be an important part of your data collection.

## 27.6 Specific Use Cases That Require Additional Tuning

This section describes some specific use cases that may benefit from additional tuning.

### 27.6.1 Oracle Access Manager Integration Tuning Parameters

The following properties and default values are used to create the Oracle Access Manager Client Object Pool. These parameters can be configured to higher values if the login volume is high. Note that any changes to these properties require a restart of the OAAM Server in order to rebuild the connection pool with new settings.

| Parameter | Description | Tuning Consideration |
|---|---|---|
| Primary OAM Server Setting:<br>oaam.uio.oam.num_of_connections | Max connections in the pool.<br>Default is 20. | Consider increasing the default value to meet your usage requirements. |

| Parameter | Description | Tuning Consideration |
|---|---|---|
| Secondary OAM Server Settings (if used):<br>oaam.uio.oam.secondary.host.num_of_connections | Max connections in the pool to secondary server. | Consider increasing the default value to meet your usage requirements. |
| oaam.oam.oamclient.minConInPool | Minimum number of connection in a pool<br>Default is 20. | Consider keeping this value the same as the Max Connections set in oaam.uio.oam.num_of_connections |
| oaam.oam.oamclient.initDelayForWatcher | Initial delay for pool watcher thread to start observing the pool.<br>Default is 300 seconds. | Specify the delay in seconds based on your requirements. |
| oaam.oam.oamclient.periodForWatcher | Pool watcher thread sleep duration in seconds.<br>Default is 300 seconds. | Keep this to low value if connections are frequently interrupted. |
| oaam.oam.oamclient.timeout | Duration of connection wait time in seconds, if no connections are available in pool.<br>Default is 60 seconds. | Keep this value to low number whenever possible. |

## 27.6.2 Oracle RAC Specific Tuning Parameters

If Oracle RAC is used to host an Oracle Adaptive Access Manager database, then consider setting the following parameters to improve Oracle Adaptive Access Manager database performance:

- Use Reverse Key Indexes for primary keys of the tables to reduce contention:

  `VCRYPT_TRACKER_USERNODE_LOGS`

  `VCRYPT_TRACKER_NODE`

  `VT_SESSION_ACTION_MAP`

- Create partitioned indexes on heavily accessed tables to reduce index contention.

## 27.6.3 SOAP Deployments

In deployments where applications are integrated with Oracle Adaptive Access Manager using SOAP (Simple Object Access Protocol) option, the following should be considered to optimize performance:

- Make sure there are no network issues between the SOAP client and Oracle Adaptive Access Manager Server
- To reduce DNS resolution issues, specify the IP Address of the Oracle Adaptive Access Manager Server where SOAP services are hosted as the value of Oracle Adaptive Access Manager Host in `vcrypt.tracker.soap.url` property

### 27.6.4 Oracle Adaptive Access Manager Offline Deployment

If you are using Oracle Adaptive Access Manager offline to perform risk evaluations on historical or non-real time login/transactional data, you may need to complete some additional configurations to ensure performance is maintained. For more information on using Oracle Adaptive Access Manager Offline, see "OAAM Offline" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

# 28

# Oracle Unified Directory Performance Tuning

This chapter provides guidelines for tuning and sizing Oracle Unified Directory. It contains these topics



> **Note:** Specific tuning information for Oracle Unified Directory is located in the "Tuning Performance" chapter of the *Oracle Unified Directory Administrator's Guide*. This chapter provides only a high-level overview.

## 28.1 About Oracle Unified Directory

Oracle Unified Directory is a comprehensive next generation directory service that is designed to address large deployments, to provide high performance, to be highly extensive and to be easy to deploy, manage, and monitor.

## 28.2 Performance Considerations

Oracle Unified Directory aims to be high-performing and highly-scalable. Although the server can achieve impressive results with the "out-of-the-box" server configuration and default JVM settings, performance can often be improved significantly through some basic tuning.

The default settings of Oracle Unified Directory are targeted at evaluators and developers who are running equipment with limited resources. When you deploy Oracle Unified Directory in a production environment, it is useful to do some initial tuning of the Java Virtual Machine (JVM) and of the server configuration to improve scalability and performance (particularly for write operations).

In addition, performance tuning strategies differ depending on whether you are running a directory server or a proxy server. This section describes some of the areas that you should consider tuning based on your server usage. Note that the specific tuning parameters and descriptions are discussed in the "Tuning Performance chapter of the *Oracle Unified Directory Administrator's Guide*.

- **When OUD is used as an LDAP Directory Server.** When used as a Directory Server, you can maximize performance by:
  - Tuning the database cache size, preload, and file cache size appropriately.

  - Placing the database on a fast file system.
  - Using the correct database caching mode for your deployment.
  - Tuning the Oracle Berkeley DB Java Edition log cleaners.
- **When OUD is used as an LDAP Proxy Server.** When used as a Proxy Server, you can maximize performance by:
  - Making sure you have a sufficient number of worker threads. Proxying requires a large number of worker threads to optimize performance.
  - Setting the heap size to an appropriate value and using the correct JVM. It is unlikely that a proxy will need more than 4GB of heap, therefore a 32-bit JVM should be used in most cases.

In addition, the following items can improve performance in specific deployment scenarios.

- **Java Version**. Use the most recent Java Runtime Environment (JRE) release available. See the Certification Matrix `<unilink:fmwcert>` to see the latest supported release of JRE.
- **Environment Variables**. The server uses the *OPENDS_JAVA_HOME* environment variable to point to your installed JRE. If you have multiple versions of Java installed on a system, set the *JAVA_HOME* environment variable to point to the root of the desired installation. In this way, the version of the JRE specified by the *JAVA_HOME* variable can be used by other applications but not by Oracle Unified Directory.

  To specify a JRE installation for the server, do one of the following:

  - Use the `dsjavaproperties` command to set the appropriate environment variables.
  - Set the *OPENDS_JAVA_BIN* environment variable (with the JAVA binary path).
  - Set the *OPENDS_JAVA_HOME* environment variable (with the JAVA installation path).

## 28.3 Monitoring Unified Directory Performance

Oracle Unified Directory provides an extensible monitoring framework. Oracle Unified Directory performance can also be monitored by using the Enterprise Manager Grid Control plugin.

For more information, see "Monitoring Oracle Unified Directory" in the *Oracle Unified Directory Administrator's Guide*.

# 29

# 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:



## 29.1 About Security Services

Oracle Fusion Middleware provides security services through Oracle Platform Security Services (OPSS) and Oracle Web Services.

- Oracle Platform Security Services

  Oracle Platform Services is a key component of Oracle Fusion Middleware. It offers an integrated suite of security services and is easily integrated with Java SE and Java EE applications that use the Java security model. Security Services includes features that implement user authentication, authorization, and delegation services that developers can integrate into their application environments. Instead of devoting resources to developing these services, application developers can focus on the presentation and business logic of their applications.

  Using Oracle Platform Security for Java, applications can enforce fine-grained access control upon resource users. The three key steps are:

  – Configure and invoke a login module, as appropriate. You can use provided login modules, or you can use custom login modules.

  – Authenticate the user attempting to log in, which is the role of the identity store service.

  – Authorize the user by checking permissions for any roles the user belongs to for whatever the user is attempting to accomplish, which is the role of the policy store service.

- Oracle Web Services Security

Oracle Web Services Security provides a framework of authorization and authentication for interacting with a web service using XML-based messages.

---

**Note:** The information in this chapter assumes that you have reviewed and understand the concepts and administration information for Oracle Fusion Middleware Security Services. For more information, see the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* before tuning any security parameters.

---

## 29.2 Detecting General Performance Issues

This section offers some general guidelines on how to identify a performance bottleneck and how to approach addressing such problems.

If you discover a performance bottleneck, you should first verify that you have addressed the expected traffic load throughout your Web services deployment. If there is a system in the critical path that is at 100% CPU usage, you may simply need to add one or more computers to the cluster.

If there is a bottleneck in your deployment, it is likely to be within one of the following:

- Traffic through a slow connection with an agent
- Latency in connections to third-party queuing systems like JMS

For any of these problems, check the following potential sources:

- Problems with policy assertions that include connections to outside resources, especially the following types:
  - Database Repositories
  - LDAP Repositories
  - Secured Resources
  - Proprietary Security Systems
- Problems with database performance

If you identify one of these as the cause of a bottleneck, you may need to change how you manage your database or LDAP connections or how you secure resources.

## 29.3 Oracle Platform Security Services Tuning

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


- JVM Tuning Parameters
- LDAP Tuning Parameters
- Authentication Tuning Parameters
- Authorization Tuning Properties
- OPSS PDP Service Tuning Parameters

**Note:** For more information on tuning your Oracle database for Oracle Platform Security Services, see the *Oracle Database Performance Tuning Guide*.

### 29.3.1 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)".

### 29.3.2 LDAP Tuning Parameters

This section covers Lightweight Directory Access Protocol (LDAP) tuning. Oracle supports the management of policies in file-based repositories: Oracle Internet Directory and Oracle Virtual Directory.

If you encounter increased CPU usage due to high SQL execution times, see the following chapters for basic tuning configurations for large deployments:

- Oracle Internet Directory configuration settings can impact performance. For more information, see Chapter 23, "Oracle Internet Directory Performance Tuning".
- In addition to being configured as a LDAP server, Oracle Virtual Directory can also be configured as a local storage adapter (LSA). See Chapter 24, "Oracle Virtual Directory Performance Tuning".

### 29.3.3 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`.

### 29.3.4 Authorization Tuning Properties

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

***Table 29–1 Authorization Properties***

| Java System Properties | Default Value | Valid Values | Notes |
|---|---|---|---|
| `-Djps.subject.cache.key` | 4 | 3<br>4<br>5 | 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.<br><br>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.<br><br>The following settings can be used to configure the cache key:<br><br>■ 3: Use the platform subject directly as the key. **Note:** On WLS if the `principalEqualCaseinsensitive` flag is enabled, two subjects could confused if their contents are the same, but the case of the principals is different.<br><br>■ 4: This setting is similar to '3' but overcomes the case-sensitive issue. This is the out-of-the-box setting.<br><br>■ 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 `WLSUSerImpl` type).<br><br>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. |

***Table 29–1 (Cont.) Authorization Properties***

| Java System Properties | Default Value | Valid Values | Notes |
|---|---|---|---|
| `-Djps.subject.cache.ttl` | 60000ms | | 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 cahed value is dumped. The setting can be controlled by the flag of `-Djps.subject.cache.ttl=xxxx`, where 'xxx' is the duration in milliseconds.<br><br>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. |
| `-Djps.combiner.optimize=true` | True | True<br>False | This system property is used to cache the protection domains for a given subject. Setting `-Djps.combiner.optimize=true` can improve Java authorization performance. |
| `-Djps.combiner.optimize.lazyeval=true` | True | True<br>False | This system property is used to evaluate a subject's protection domain when a checkPermission occurs. Setting `-Djps.combiner.optimize.lazyeval=true` can improve Java authorization performance. |
| `-Djps.policystore.hybrid.mode=true` | True | True<br>False | This 'hybrid mode' property is used to facilitate transition from SUN java.security.Policy to OPSS Java Policy Provider.<br><br>The OPSS Java Policy Provider reads from both java.policy and system-jazn-data.xml."Hybrid" mode can be disabled by setting the system property `jps.policystore.hybrid.mode` to `false` when starting the WebLogic Server. Setting `-Djps.policystore.hybrid.mode=false` can reduce runtime overhead. |
| `-Djps.authz=ACC` | ACC | ACC<br>SM | Delegates the call to JDK API `AccessController.checkPermission` which can reduce the performance impact at run time or while debugging.<br><br>ACC: delegate to `AccessController.checkPermission`<br><br>SM: delegate to `SecurityManager` if SecurityManager is set. |

### 29.3.5 OPSS PDP Service Tuning Parameters

Table 29–2 provides OPSS tuning parameters for policy store. The following properties should only be used for OPSS PDP services running in non-controlled distribution mode. The properties should be defined for OPSS PDP service (or policystore service if PDP service absent):

***Table 29–2 OPSS PDP Service Tuning Parameters***

| Parameter | Default Value | Valid Values | Notes |
|---|---|---|---|
| `oracle.security.jps.policystore.rolemember.cache.type` | STATIC | STATIC, SOFT, WEAK | This parameter specifies the type of role member cache.Valid only in Java EE applications.<br>Valid values:<br>■ STATIC: Cache objects are statically cached and can be cleaned explicitly only according the applied cache strategy, such as FIFO. The garbage collector does not clean a cache of this type.<br>■ SOFT: The cleaning of a cache of this type relies on the garbage collector when there is a memory crunch.<br>■ 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.<br>Consider maintaining the default value for the best performance. |
| `oracle.security.jps.policystore.rolemember.cache.strategy` | FIFO | FIFO<br>NONE | The type of strategy used in the role member cache. Valid only in Java EE applications.<br>Valid values:<br>■ FIFO: The cache implements the first-in-first-out strategy.<br>■ 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.<br>Consider maintaining the default value for the best performance. |
| `oracle.security.jps.policystore.rolemember.cache.size` | 1000 | | 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. |

***Table 29–2 (Cont.) OPSS PDP Service Tuning Parameters***

| Parameter | Default Value | Valid Values | Notes |
|---|---|---|---|
| `oracle.security.jps.policystore.policy.lazy.load.enable` | True | True<br>False | 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. |
| `oracle.security.jps.policystore.policy.cache.strategy` | PERMISSION_FIFO | PERMISSION_FIFO<br>NONE | The type of strategy used in the permission cache. Valid only in Java EE applications.<br>Valid Values:<br>■ PERMISSION_FIFO: The cache implements the first-in-first-out strategy.<br>■ 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.<br>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<br>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<br>False | This property is used for refresh enabling. Consider maintaining the default value for performance. |
| `oracle.security.jps.policystore.refresh.purge.timeout` | 43200000 | | 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<br>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).<br>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. |

## 29.4 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.



### 29.4.1 Choosing the Right Policy

Oracle Web Services Security supports many policies and the appropriate policies must be implemented based on the security need of the deployment. Careful consideration should be given to performance, since each additional policy can impact performance. For example Transport level security (SSL) is faster than Application level security, but transport level security can be vulnerable in multi-step transactions. Application level security has more performance implications, but provides end-to-end security.

See "Configuring Policies" in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* to determine which security policies are required for a deployment.

### 29.4.2 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.

### 29.4.3 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.

### 29.4.4 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"
```

### 29.4.5 Monitoring the Performance of Web Services

You can monitor the performance on the following Oracle Web Services through the Web Services home page of Oracle Fusion Middleware Control:

- Endpoint Enabled Metrics such as:
  - Policy Reference Status
  - Total Violations
  - Security Violations
- Invocations Completed
- Response Time, in seconds
- Policy Violations such as:
  - Total Violations
  - Authentication Violations
  - Authorization Violations
  - Confidentiality Violations
  - Integrity Violations
- Total Faults

For general information on monitoring Oracle Fusion Middleware components, see Chapter 4, "Monitoring Oracle Fusion Middleware".

For detailed information on using Oracle Fusion Middleware Control to monitor Oracle Web Services, see "Monitoring the Performance of Web Services" in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

# 30

# Oracle Entitlements Server Performance Tuning

This chapter provides guidelines for tuning and sizing Oracle Entitlements Server (OES). It contains these topics:

- Section 30.1, "About Oracle Entitlements Server"
- Section 30.2, "Performance Considerations for Oracle Entitlements Server"
- Section 30.3, "Basic Tuning Considerations"

> **Note:** As with any enterprise class business application, there is no simple procedure for tuning that works for all systems. The tuning sections in this chapter provide (in some cases) sample configurations and outline the principles for tuning Oracle Identity Manager. Consider your own use case scenarios to determine which settings are appropriate.

## 30.1 About Oracle Entitlements Server

Oracle Entitlements Server (OES) is a standards-based, policy-driven security solution that provides real time fine-grained authorization in Application, Service-Oriented Architecture (SOA) and Database environments.

Oracle Entitlements Server allows an organization to protect its resources by defining and managing policies that control access to, and usage of, these resources. Access privileges are defined in a policy by specifying who can do what to which resource, when it can be done, and how. The policy can enforce controls on all types of resources including software components (URLs, Java Server Pages, Enterprise JavaBeans, methods, servlets and the like used to construct an application) and business objects (representations of user accounts, personal profiles and contracts such as bank accounts in a banking application, patient records in a health care application, or anything used to define a business relationship).

For more information on using Oracle Entitlements Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle Entitlements Server*.

## 30.2 Performance Considerations for Oracle Entitlements Server

Effective Oracle Entitlement Server performance tuning starts with a good understanding of its usage and general performance issues. Before you begin tuning Oracle Entitlement Server performance, review this section as well as the recommendations discussed in Chapter 2, "Top Performance Areas".

| | |
|---|---|
| Number of users | Understanding the overall user population size; group, membership, attribute counts, Grantees, Resource, permissions, entitlement & policies; data types, and configuration parameters of the LDAP and database is essential.<br><br>See Chapter 3, "Performance Planning" for more information on using population data to improve performance. |
| Daily activity usage | It is important to know how many users are active during a 24-hour period and the expected traffic. Spikes in usage may require additional tuning to avoid performance issues.<br><br>See Chapter 4, "Monitoring Oracle Fusion Middleware" for more information on collecting performance data. |
| Hardware resources and topology | Like any application deployed in demanding, real-time environments, proper server sizing and configuration is critical for acceptable Oracle Entitlements Server performance. Ensuring that your hardware is sufficient to prevent bottlenecks is a key factor in performance tuning Oracle Entitlements Server.<br><br>See "Securing Sufficient Hardware Resources" for more information on optimizing hardware resources. |
| JVM and garbage collection | Optimal performance of Oracle Entitlements Server depends on correctly tuning JVM heap sizes and garbage collection.<br><br>See "Configuring Garbage Collection" and "Specifying Heap Size Values" for more information. |
| Tuning the OES Caches | Understanding the OES caches (Decision cache and Permission cache) and tuning the same is important to achieve the optimal performance. |

## 30.3 Basic Tuning Considerations

Depending on your OES usage and performance issues, you may consider tuning the following basic parameters. See Chapter 2, "Top Performance Areas" for additional tuning considerations.

This section includes the following topics:

- Tuning the OES Policy Store
- Tuning of OES Administration Server
- Performance Tuning OES Security Modules

### 30.3.1 Tuning the OES Policy Store

OES policy store is the first component that needs to be tuned from the overall performance tuning. The tuning policy store involves tuning of the underlying database and the EclipseLink. The following sections provide information about the specific tuning aspects as a part of policy store tuning.

This section includes the following topics:

- Oracle Database System Parameters Tuning
- Table Spaces Tuning During User Schema Creation (RCU)
- Tuning the OES Schema
- EclipseLink Tuning

#### 30.3.1.1 Oracle Database System Parameters Tuning

In addition to the standard database tuning parameters described in Section 2.6, "Tuning Database Parameters", the properties listed in Table 30–1 should also be tuned when using OES.

***Table 30–1 Oracle Database System Parameters***

| Property Name | Value |
|---|---|
| `Processes` | 1500 |
| `sga_target` | 3221225472 (3GB) |
| `audit_trail` | none |
| `open_cursors` | 500 |
| `pga_aggregate_target` | 1610612736 (1.5GB) |
| `nls_sort` | BINARY |
| `filesystemio_options` | SETALL |
| `fast_start_mttr_target` | 3600 |
| `db_securefile` | ALWAYS |
| `session_cached_cursors` | 500 |
| `plsql_code_type` | NATIVE |
| `"_b_tree_bitmap_plans"` | False |
| `Memory_target` | 0 |
| `kernel.shmall`, `kernel.shmmax`, `kernel.sem` and `fs.file-max` | You must ensure that `kernel.shmall`, `kernel.shmmax`, `kernel.sem` and `fs.file-max` are set at optimal values (all kernel level parameter can be tuned and obtained from `/etc/sysctl.conf`.) |

**Note:** The above values are provided to show the common tuning configurations. There are exceptions and constraints. For example, SGA, PGA sizes are limited by the underlying operating system restrictions on the maximum available memory; on some platforms the processes can be calculated based on the connection pool settings. You must ensure that the tunings are validated by DBA's.

For more information about Oracle Database Tunings, see Section 2.7, "Reusing Database Connections".

#### 30.3.1.2 Table Spaces Tuning During User Schema Creation (RCU)

The OES DB schema is created via RCU (Repository Creation Utility). To create a database schema using RCU, do the following:

- Allocate large disk spaces for OES table spaces for optimal performances. Use 6GB files for `IAS_OPSS` and `IAS_TEMP` table spaces.
- If you want to manage many large applications in the policy store, allocate huge table space size, for example, 40 GB for `IAS_OPSS` table space, 15 GB for `IAS_TEMP` table space.
- To add applications to the policy store, mark the table space increment as 1GB, instead of 100MB, to avoid continuous extending of table space when it is full.

#### 30.3.1.3 Tuning the OES Schema

The required OES database (DB) schema tuning should be complete if you have followed the steps mentioned in the above sections. Running the DB stats as the OES DB schema user is an on-going practice that must be monitored, because the policy data is stored in the DB.

Schema statistics should not be gathered for column `JPS_DN.PARENTDN`. Gathering histogram for column `JPS_DN.PARENTDN` impacts the query performance. To disable histogram, use the following SQL script:

```
EXECUTE dbms_stats.delete_column_stats(ownname=>'<SCHEMA_NAME>',
tabname=>'JPS_DN', colname=>'PARENTDN', col_stat_type=>'HISTOGRAM');
```

Run the SQL script only once.

Run the command `DBMS_STATS.gather_schema_stats('<SCHEMA_NAME>',DBMS_STATS.AUTO_SAMPLE_SIZE,no_invalidate=>FALSE)`

You can automate the command by defining `DB job` to run the `dB stats` with required frequency. This collection must be done after a large policy data migration into the store.

#### 30.3.1.4 EclipseLink Tuning

OES uses `EclipseLink` as JPA (Java Persistence API) implementation for Database operations. Table 30–2 defines the `EclipseLink` tuning parameters that can be set through `jps-config.xml`.

***Table 30–2 EclipseLink Tuning***

| Property | Description | Default |
|---|---|---|
| `eclipselink.jdbc.connection_pool.default.max` | the maximum number of read connection in the internal connection pool | 32 |
| `eclipselink.jdbc.connection_pool.default.min` | the minimum number of read connection in the internal connection pool | 32 |

The default property values can be overwritten by setting DB policy store service instance for OES admin, or PDP service in controlled-pull mode, as in the following:

```
      <propertySet name="props.db.1">
       <property name="jdbc.url"
       value="jdbc:oracle:thin:@slc04jpa.example.com:1521/orcl2"/>
       <property name="oracle.security.jps.farm.name" value="cn=my_domain"/>
       <property name="server.type" value="DB_ORACLE"/>
... ...

       <property name=" eclipselink.jdbc.connection_pool.default.min" value="16"/>
        <property name=" eclipselink.jdbc.connection_pool.default.min"
value="64"/>
</propertySet>
... ...

<serviceInstance name="policystore.db" provider="policystore.provider">
       <property name="policystore.type" value="DB_ORACLE"/>
       <propertySetRef ref="props.db.1"/>
</serviceInstance>
```

For more information on EclipseLink properties, see `http://www.eclipse.org/eclipselink/api/2.3/org/eclipse/persistence/config/PersistenceUnitProperties.html`.

## 30.3.2 Tuning of OES Administration Server

This section provides information on how to tune the JVM parameters for optimal performance on large policy data for operations, like migration and distribution.

This section includes the following topics:

- WLST Tuning
- OES Admin Server Tuning

### 30.3.2.1 WLST Tuning

WebLogic Scripting Tool (WLST) is a command-line scripting interface of Oracle Middleware. It is used for OES for some management work, including migration of security store. For migrating large applications, you must tune the JVM parameters for the WLST tool for Java memory size, as in the following:

```
Sun:
"-Xms6144m -Xmx6144m -Xmn1900m -XX:PermSize=128m -XX:MaxPermSize=1024m"

JRockit:
"-Xms6144m -Xmx6144m -Xns1900m"
```

Note that the required heap size parameter (`ms, mx, mn`) could be based on the size of application you migrate. Table 30–3 provides a sample data:

***Table 30–3 Data sample***

| Application | XML2DB | DB2XML | Number of Policies |
|---|---|---|---|
| Sample1KPolicyApp | 650MB | 1800 MB | 1132 |
| Sample5KPolicyApp | 2.7 GB | 6 GB | 5351 |

### 30.3.2.2 OES Admin Server Tuning

OES Admin Server must be tuned for optimized performance for the policy distribution of large applications . The following are the tuning guidelines for OES Admin Server:

- Use JDK version 1.6.0_31 or higher.
- The graphic below provides information about the JDK memory size tuning. A large memory is required on Admin Server to generate application policy snapshot.

```
Sun:
"-Xms6144m -Xmx6144m -Xmn1900m -XX:PermSize=128m -XX:MaxPermSize=1024m"

JRockit:
"-Xms6144m -Xmx6144m -Xns1900m"
```

## 30.3.3 Performance Tuning OES Security Modules

When OES Security Module is configured as controlled pull mode, it has two main functionalities:

- Acts as a PDP and/or PEP- it receives an authorization request, makes decision and enforces the decision.
- Calls Policy Distribution component periodically to make configured policies and policy data available for evaluation.

When tuning the OES Security Module ensure that there is high throughput of authorization request processing and distribute the policy efficiently.

The size of the Policy data affects performance of security module. The number of functions, attributes, roles, resources, and policies have a great influence on authorization throughput. Distribution time and memory size is required for the Security Module.

For example:

```
Application Name:  myapp
oracle.security.jps.ldap.root.name:  cn=jpsroot
oracle.security.jps.farm.name:  cn=base_domain
```

***Table 30–4 Security Module Tuning Attributes***

| Item | Number | Comment | Query SQL |
|---|---|---|---|
| Resource Type | 1 | | |
| Attribute definition | 17 | | |
| Function definition | 269 | | select count (*) from jps_dn where parentdn like ''cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,%cn=functions,'; |
| Application Role | 0 | No application role is created. Weblogic Server's roles and principals are used. | |
| Grantee | 5351 | Principals of policy are stored as grantee, one policy for one principal. | select count (*) from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,cn=jaas policy,cn=grantees,'; |
| Permission | 1772 | Resource action pairs used in policies. | select count (*) from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,cn=jaas policy,cn=permissions,'; |
| Resource | 3173 | | select count (*) from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,%cn=resources,'; |
| Entitlement | 1806 | | select count (*) from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,%cn=permission sets,'; |
| Policy | 5351 | | select count (*) from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,%cn=policies,'; |

*Table 30–4 (Cont.) Security Module Tuning Attributes*

| Item | Number | Comment | Query SQL |
|---|---|---|---|
| Rule | 5351 | | select count (*) from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,%cn=rules,'; |
| Attributes | 1,484,356 | Attributes of above entries | select count (*) from jps_attrs where jps_dn_entryid in (select entryid from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,%'); |
| assignee attributes | 1,220,877 | Attributes for policy assignments | select count (*) from ct_9_1 where jps_dn_entryid in (select entryid from jps_dn where parentdn like 'cn=jpsroot,cn=jpscontext,cn=base_domain,cn=myapp,cn=jaas policy,cn=permissions,'); |

For the above policy data sample runtime cache memory size can be computed as shown in Table 30–5

*Table 30–5 Sample runtime cache memory size calculations*

| Cache Type | Total Heap Size | Formula |
|---|---|---|
| Resource Policy Cache | 476.91M | 0.4K * Assignee number |
| Permission Cache | 143.78M | 0.12K * Assignee number + 0.41K * Permission Number |
| Resource Cache | 0.13M | 44 * Resource number |
| Policy Cache | 1.46MB | 286 * Policy number |
| Resource Type Cache | Small one, ignore them | |
| Function Cache | Small one, ignore them | |
| Attribute Cache | Small one, ignore them | |

For a better performance of large data operations, a total of 6GB heap size is required for Security Module.

**JVM Tunings for Security Module: (WLS/Tomcat/Java)**

Sun JVM:

```
"-Xms6144m -Xmx6144m -Xmn1900m -XX:PermSize=128m -XX:MaxPermSize=1024m
-XX:+UseParallelGC "
```

`-XX :+UseParallelGC`: Use parallel garbage collection for scavenges. This collection algorithm is designed to maximize throughput while minimizing pauses and a suitable GC algorithm for Security Module.

JRockit:

```
"-Xms6144m -Xmx6144m -Xns1900m"
```

This section includes the following topics:

- "Tuning OES Distribution Service"
- "Tuning OES Cache"

- "Network Considerations"
- "Resource Intensive Operations"
- "Enable Logging for Performance Measurement"

#### 30.3.3.1 Tuning OES Distribution Service

**PDP service configuration**

`oracle.security.jps.pd.client.PollingTimerInterval`:

This property specifies PD periodic check interval. The default value is 600 seconds. Depending on how often policy data is changed, it can be increased or decreased as needed.

**PDP service configurations for WSSM:**

Table 30–6 lists the available parameters for WS SM

***Table 30–6 Available Parameter for WSSM***

| Property | Description |
|---|---|
| `oracle.security.jps.pdp.sm.IdentityCacheEnabled` | Flag to set if using identity cache. If not set, identity cache is enabled by default |
| `oracle.security.jps.pdp.sm.IdentityMaxCacheSize` | Specify the maximum cache size.<br>The default value is 20000.<br>Tune this parameter according to the number of all subjects. |
| `oracle.security.jps.pdp.sm.IdentityCacheEvictionPercentage` | Specifies percentage of identities that must be evicted when cache has reached the maximum size. The default value is 20 percentage. |
| `oracle.security.jps.pdp.sm.IdentityCachedEntryTTL` | Specifies time-to-live in seconds for a identity record in the cache.<br>The default value is 3600 seconds. |
| `oracle.security.jps.pdp.wssm.WSLoggingSoapHandlerEnabled` | Flag to set if enable `EnvelopLoggingSOAPHandler` of ws sm.<br>The default value is false. |

#### 30.3.3.2 Tuning OES Cache

This section describes the decision cache attributes of the Security Module.

Set the following properties in the `jps-config.xml` of PDP:

```
<property name="oracle.security.jps.pdp.AuthorizationDecisionCacheEnabled"
value="true"/>
<property
name="oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionCapacity"
value="1500"/>
<property
name="oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionPercentage"
value="10"/>
<property name="oracle.security.jps.pdp.AuthorizationPerUserDecisionCacheSize"
value="1500"/>
<property name="oracle.security.jps.pdp.AuthorizationDecisionCacheTTL"
value="300"/>
```

*Table 30–7 Introduction to the Properties*

| Property Name | Description | Accepted Values | Optional/ Mandatory | Comments |
|---|---|---|---|---|
| `oracle.security.jps.pdp.AuthorizationDecisionCacheEnable` | Specifies whether the the policy decision cache should be enabled. | "true/false"<br>Default: true | Optional | Since DW PS1 |
| `oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionCapacity` | This is the number that is used to evict the decision cache if the decision cache size reaches this size. | Number<br>Default:500 | Optional | Since DW PS1 |
| `oracle.security.jps.pdp.AuthorizationDecisionCacheEvictionPercentage` | The percentage of authorization decisions to drop when the decision cache has reached the maximum capacity. | Number<br>Default:10 | Optional | Since DW PS1 |
| `oracle.security.jps.pdp.AuthorizationDecisionCacheTTL` | This is the TTL for the Decision Cache in seconds | Number<br>Default:60 | Optional | Since DW PS1 |
| `oracle.security.jps.pdp.AuthorizationPerUserDecisionCacheSize` | This is the number that is used to evict the decision cache of each user (Subject) if the second level decision cache size reaches this size. | Number<br>Default:1000 | Optional | Since DW PS2 |

#### 30.3.3.3 Network Considerations

The performance of the overall network is a major factor in the performance of the system. A reduction in network latency can improve network performance.

To control network latency, consider the following:

- Keep database repositories close to the OES servers. Installing OES servers on a remote server may cause significant latency. Latency between the application tier and the database tier should be 5ms or less to maintain optimal performance.
- Add an SSL accelerator or load balancer outside of the Oracle Entitlement Server system to improve the performance of your network.
- Deploying a load balancer in front of the Web servers or application servers is a best practice for increasing availability and performance of Web-based applications, including Oracle Entitlement Server. However, load balancers are not recommended between the Oracle Entitlement Server components themselves.
- Place the OES Servers closer to client applications than to the Oracle Internet Directory (OID) server.

#### 30.3.3.4 Resource Intensive Operations

Optimization was made for process of `PepRequestFactory.newQueryPepRequest().decide()`, that does authorization check for a bunch of resources.

This feature is disabled by default and should be enabled for SM instances in controlled distribution mode.

To enable it, you should update `jps-config.xml` of SM instances. For a PDP service instance, add the following property.

```
<serviceInstance name="pdp.service" provider="pdp.service.provider">
        ... ...
        property name="oracle.security.jps.runtime.enableResourcePermissionCache"
value="true"/>
```

### 30.3.3.5 Enable Logging for Performance Measurement

Logs are required for performance measurement for some key operations, including migration, snapshot generation and distribution.

There is log in INFO level for migration via WLST. You could see logs with timestamp print to WLST console directly, as shown in the following:

```
Apr 19, 2013 3:07:00 AM
oracle.security.jps.internal.tools.utility.JpsUtilMigrationPolicyImpl
migrateAppPolicyData
INFO: Migration of Application Policies in progress.....

... ...
Apr 19, 2013 3:07:39 AM
oracle.security.jps.internal.tools.utility.util.JpsMigrateUtil cloneResourceType
INFO: Migration of Resources started
Apr 19, 2013 3:08:12 AM
oracle.security.jps.internal.tools.utility.util.JpsMigrateUtil cloneResourceType
INFO: Migration of Resources completed in 00:00:33

... ...
Apr 19, 2013 3:08:44 AM
oracle.security.jps.internal.tools.utility.util.JpsMigrateUtil clonePermissionSet
INFO: Migration of Permission Sets completed in 00:00:31

... ...
Apr 19, 2013 3:11:35 AM
oracle.security.jps.internal.tools.utility.destination.apibased.JpsDstPolicy clone
INFO: Completed Migrating 5,351 policies in [170,914] ms.

... ...
Apr 19, 2013 3:20:57 AM
oracle.security.jps.internal.tools.utility.destination.apibased.JpsDstPolicy clone
INFO: Migration of Grants completed in 00:09:22

... ...
Apr 19, 2013 3:20:57 AM
oracle.security.jps.internal.tools.utility.JpsUtilMigrationPolicyImpl
migrateAppPolicyData
INFO: Migration of Application Policies completed, Time taken for migration is
00:13:57
```

#### Enable Logging for Snapshot Generation Measurement

The following are the logging properties that need to be enabled to measure the timing for snapshot generation as a part of policy distribution. Create Java `logging.properties` file with log level setting as shown in the following:

```
handlers= java.util.logging.FileHandler
.level= INFO
############################################################
# Handler specific properties.
# Describes specific configuration info for Handlers.
```

```
############################################################ default file output
is in user's home directory
java.util.logging.FileHandler.pattern =<Log_home>/performance.log
java.util.logging.FileHandler.limit = 10000000
java.util.logging.FileHandler.count = 200
java.util.logging.FileHandler.formatter =java.util.logging.SimpleFormatter
java.util.logging.FileHandler.level = FINE
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
############################################################
# Facility specific properties.
############################################################
# Provides extra control for each logger.
############################################################
# For example, set the com.xyz.foo logger to only log SEVERE
# messages:
oracle.security.jps.internal.policystore.SnapshotWorker.level=FINEST
```

Enable log in <DOMAIN_HOME>/bin/startWeblogic.sh

```
${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS}
-Djava.util.logging.config.file=./logging.properties
-Dweblogic.Name=${SERVER_NAME} -Djava.security.policy=${WL_
HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS} ${PROXY_SETTINGS}
${SERVER_CLASS}
```

From the log file, you should see the following log message when you generate a snapshot of application policy:

```
Jan 31, 2013 1:07:49 AM oracle.security.jps.internal.policystore.SnapshotWorker
run
FINE:SnapshotWorker begin.
..........
Jan 31, 2013 1:12:43 AM oracle.security.jps.internal.policystore.SnapshotWorker
run
FINE: SnapshotWorker end.
```

### Enable Logging for Distribution

The following are the logging properties that need to be enabled to measure the timing for snapshot generation as a part of policy distribution. Create a `logging.properties` file and add the following line:

```
oracle.security.jps.az.internal.runtime.pd.receiver.UpdatePolicySet.level=FINEST
```

You must update different script for different Security Module type.

#### Java SM

Edit the java SM start scripts to add log properties.

Take<OES_CLIENT_HOME>/oes_sm_instances/<SM_NAME>/run-j2se.sh as example.

```
${JAVA_HOME}/bin/java -Doracle.security.jps.config=${OES_INSTANCE_
HOME}/config/jps-config.xml
${MEM_ARGS} -Djava.util.logging.config.file=./config/logging.properties
oracle.security.oes.tools.OESJ2SESampleApp PD
```

And enable
`oracle.security.jps.az.internal.runtime.pd.receiver.UpdatePolicySet.level=FINEST` in the log property file.

#### WS SM

In <OESCLIENT>/oes_sm_instances/<wssm>/startWSServer.sh add following line:

```
JAVA_OPT="$JAVA_OPT
-Djava.util.logging.config.file=./config/logging.properties".
```

**Weblogic SM**

Edit file <DOMAIN_HOME>/bin/startWeblogic.sh in the end:

```
${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS}
-Djava.util.logging.config.file=./config/logging.properties
```

**Tomcat SM**

Edit file <TOMCAT_HOME>/ bin/catalina.sh, and add the following line after the last JAVA_OPTS line:

```
JAVA_OPTS="$JAVA_OPTS
-Djava.util.logging.config.file=./config/logging.properties".

Jan 31, 2013 10:11:19 PM
oracle.security.jps.az.internal.runtime.pd.receiver.UpdatePolicySet commit
FINE: Starting ...

...........
Jan 31, 2013 10:20:06 PM
oracle.security.jps.az.internal.runtime.pd.receiver.UpdatePolicySet commit
FINE: completed. Active Apps: [[]], Bounded Apps [[fidelity5000]
Jan 31, 2013 10:20:07 PM
oracle.security.jps.az.internal.runtime.pd.receiver.UpdatePolicySet commit
FINE: afterCommit is finished
Jan 31, 2013 10:20:07 PM
oracle.security.jps.az.internal.runtime.pd.receiver.UpdatePolicySet
finishFirstDist
```

# Part VI

# Oracle WebCenter Components

This part describes configuring Oracle WebCenter components to improve performance. It contains the following chapter:

- Chapter 31, "Oracle WebCenter Portal Performance Tuning"

# 31

# Oracle WebCenter Portal Performance Tuning

This is a chapter summary element.

This chapter outlines how to tune configuration properties for the operating system on which WebCenter Portal applications are installed, WebCenter Portal applications, and their back-end components.



## 31.1 About Oracle WebCenter Portal

Oracle WebCenter Portal 11*g* is an integrated suite of products used to create social applications, enterprise portals, communities, composite applications, and internet or intranet Web sites on a standards-based, service-oriented architecture (SOA). Oracle WebCenter Portal combines the development of rich internet applications, a multi-channel portal framework, and a suite of horizontal Enterprise 2.0 applications, which provide content, presence, and social networking capabilities to create a highly interactive user experience. Interacting with services such as instant messaging, blogs, wikis, RSS, tags, discussion forums, activities and social networks directly within the context of a portal or an application improves user and group productivity and enhances the return on IT investments.

Oracle WebCenter Portal: Spaces is an out-of-the-box WebCenter Portal application that brings you the latest technology in terms of social networking, communication, collaboration, and personal productivity with no development effort. Through the robust set of integrated services and applications provided by Oracle WebCenter Portal's Framework, Composer, and Resource Catalog, the Spaces application enables you to deploy instant community portals, team sites and other collaborative applications.

For more information about Oracle WebCenter Portal, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal* and *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

## 31.2 Basic Tuning Considerations

The tuning considerations in this section apply to most WebCenter Portal and WebCenter Portal: Spaces deployments and usage scenarios. It is highly recommended that you review these configurations and implement those that are appropriate for your use case scenarios.

### 31.2.1 Setting System Limit

To run a WebCenter Portal application at moderate load, set the `open-files-limit` to `4096`. If you encounter errors, such as **running out of file descriptors**, then increase the system limit.

For example, on Linux, you can use this command:

```
ulimit -n 8192
```

Refer to your operating system documentation to find out how to change this system limit.

### 31.2.2 Setting JDBC Data Source

The following data source settings are the defaults for `mds-SpacesDS` and `WebCenterDS`. These settings can be adjusted depending on the application's usage pattern and load.

```
    <jdbc-connection-pool-params>
        <initial-capacity>10</initial-capacity>
        <max-capacity>50</max-capacity>
        <capacity-increment>1</capacity-increment>
        <shrink-frequency-seconds>0</shrink-frequency-seconds>
        <highest-num-waiters>2147483647</highest-num-waiters>

<connection-creation-retry-frequency-seconds>0</connection-creation-retry-frequenc
y-seconds>

<connection-reserve-timeout-seconds>60</connection-reserve-timeout-seconds>
        <test-frequency-seconds>0</test-frequency-seconds>
        <test-connections-on-reserve>true</test-connections-on-reserve>

<ignore-in-use-connections-enabled>true</ignore-in-use-connections-enabled>

<inactive-connection-timeout-seconds>0</inactive-connection-timeout-seconds>
        <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
        <login-delay-seconds>0</login-delay-seconds>
        <statement-cache-size>5</statement-cache-size>
        <statement-cache-type>LRU</statement-cache-type>
        <remove-infected-connections>true</remove-infected-connections>

<seconds-to-trust-an-idle-pool-connection>60</seconds-to-trust-an-idle-pool-connec
tion>
        <statement-timeout>-1</statement-timeout>
        <pinned-to-thread>false</pinned-to-thread>
    </jdbc-connection-pool-params>
```

To edit JDBC data source settings:

1. Login to WebLogic Server Administration Console.
2. From the Home page, select **Summary of JDBC Data Sources**, **Settings for mds-SpacesDS**, and then the **Connection Pool** tab.

3. Edit properties, as required.

To edit WebCenter Portal data source settings:

1. Login to WebLogic Server Administration Console.
2. From the Home page, select **Summary of JDBC Data Sources** and navigate to the **Connection Pool** tab.

See also "Tuning Data Source Connection Pools" in *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server*.

### 31.2.3 Setting JRockit Virtual Machine (JVM) Arguments

JVM arguments are set in the `setDomainEnv.sh` file on Unix operating systems and `setDomainEnv.cmd` on Windows operating systems. The `setDomainEnv` file is located in the <*domain_dir*>/bin directory.

> **See Also:** Section 2.4, "Tuning Java Virtual Machines (JVMs)"

- **WebLogic Server production mode:** When Webcenter is installed for production deployment, the WebLogic Server is set to production mode. However, if it is installed for development and then switched to production mode for better performance, you need to include the following parameter in the startup command:

  ```
  -Dweblogic.ProductionModeEnabled=true
  ```

  For information on setting your domain to production mode using the Administration Console, see "Change to production mode" in the Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help.

- **Heap size:** If the server is overloaded, that is, garbage is collected or out of memory error occurs frequently, then increase the heap size as appropriate to your server's available physical memory.

  For more information, see Section 2.4.1.1, "Specifying Heap Size Values" and "Set Java options for servers started by Node Manager" in the Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help.

  The following parameters can be modified in the server's startup command or through the Administration Console to increase heap size:

  ```
  jrockit vm: -Xms2048M -Xmx2048M -Xns512M

  hotspot vm: -Xms2048M -Xmx2048M -XX:MaxPermSize512M
  ```

### 31.2.4 Using Content Compression to Reduce Downloads

If clients connect to your server using relatively slow connections, that is, using modems or VPN from remote locations, consider compressing content before it downloads to the client. While content compression increases the load on the server, the client's download experience is much improved.

Several content compression methods are available. The following steps describe how to use the `mod_deflate` module from Apache.

1. Enable `mod_deflate` module on Apache.

To do this, add the following to `httpd.conf ($OH/instances/$INSTANCE_NAME/config/OHS/$OHS_NAME)`

```
LoadModule deflate_module "${ORACLE_HOME}/ohs/modules/mod_deflate.so"
```

2. Setup the Output Filter and specify the rules for compression.

   Here is a sample snippet that you can add to the `httpd.conf` (same location mentioned above). Modify the content based on your content and the compression requirements.

```
<IfModule mod_deflate.c>
SetOutputFilter DEFLATE
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/xml
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/atom+xml
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE text/html
SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.(?:exe|t?gz|zip|bz2|sit|rar)$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.(?:pdf|doc?x|ppt?x|xls?x)$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.avi$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.mov$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.mp3$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.mp4$ no-gzip dont-vary
</IfModule>
```

For more information about `mod_deflate`, refer to:
http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

## 31.3 Tuning WebCenter Portal Application Configuration

This section describes parameters that enable administrators to tune performance of WebCenter Portal applications.

This section includes the following:

- Setting Session Timeout for a Spaces Application
- Setting HTTP Session Timeout
- Setting JSP Page Timeout
- Setting ADF Client State Token
- Setting ADF View State Compression
- Setting MDS Cache Size and Purge Rate
- Configuring Concurrency Management

### 31.3.1 Setting Session Timeout for a Spaces Application

The default session timeout for a Spaces application is derived from the HTTP session timeout specified in web.xml. The out-of-the-box web.xml setting for `<session-timeout>` is 45 minutes. See Setting HTTP Session Timeout.

Administrators can use the `wcSessionTimeoutPeriod` attribute in webcenter-config.xml to increase or decrease the session timeout if required. See also "webcenter-config.xml" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

### 31.3.2 Setting HTTP Session Timeout

To manage overall resource usage, adjust the application's http session timeout value, in minutes, in the `web.xml` file. In general, shorter session timeout values correspond to less memory and CPU usage on the server.

If you must modify this property, post deployment, you must edit `web.xml` manually. See "Editing web.xml Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `web.xml`:

```
<session-config>
       <session-timeout>
          45
       </session-timeout>
</session-config>
```

### 31.3.3 Setting JSP Page Timeout

You can specify an integer value, in seconds, after which any JSP page will be removed from memory if it has not been requested in the `web.xml` file. This frees up resources in situations where some pages are called infrequently.

Increasing the value reduces user response time, and decreasing it reduces application memory foot print. The default value for is 600 seconds or 10 minutes. If jsp_timeout is not specified, it means there is no timeout.

To modify this property post deployment, you must edit `web.xml` manually. See "Editing web.xml Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `web.xml`:

```
<servlet>
         <servlet-name>
            oraclejsp
        <init-param>
            <param-name>
               jsp_timeout
            </param-name>
            <param-value>
               600
            </param-value>
       </init-param>
```

### 31.3.4 Setting ADF Client State Token

Through this setting, you can control the number of pages users can navigate using the browser Back button without losing page state. To reduce CPU and memory usage, you can decrease the value in the `web.xml` file.

If you must modify this property, post deployment, you must edit `web.xml` manually. See "Editing web.xml Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample code snippet of `web.xml`:

```
<context-param>
       <param-name>
          org.apache.myfaces.trinidad.CLIENT_STATE_MAX_TOKENS
       </param-name>
       <param-value>
          3
       </param-value>
</context-param>
```

### 31.3.5 Setting ADF View State Compression

Through this setting, you can control ADF View State Compression. By default this setting is enabled (parameter value is 'True') and all non-current view states are compressed before saving in memory, which reduces the heap usage.

Though not recommended for WebCenter Portal, you can disable this property by editing the web.xml manually. See "Editing web.xml Properties" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

The following is a sample code snippet of `web.xml`:

```
<context-param>
       <param-name> org.apache.myfaces.trinidad.COMPRESS_VIEW_STATE
      </param-name>
       <param-value> false </param-value>
</context-param>
```

### 31.3.6 Setting MDS Cache Size and Purge Rate

The default MDS cache size is 100MB. If you encounter the error message, **JOC region full**, then you can increase the MDS cache size in the `adf-config.xml` file.

Post deployment, modify these properties through the System MBeans Browser. For more information, see the section "Changing MDS Configuration Attributes for Deployed Applications" in *Oracle Fusion Middleware Administrator's Guide*.

The following is a sample snippet of `adf-config.xml`:

```
<cache-config>
<max-size-kb>150000</max-size-kb>
</cache-config>
```

Consider setting the MDS `auto-purge seconds-to-live` parameter (as shown in the example below) to remove older versions of metadata automatically every hour. By default the `auto-purge seconds-to-live="-1"` which means no purge. However, if the WebCenter Portal site changes frequently, such as when creating or editing spaces or pages, then auto-purge should be set to an appropriate value to remove older version of metadata for optimal performance.

If excessive metadata is accumulated and each purge is very expensive, reduce this interval in the `adf-config.xml` file.

The following is a sample snippet of `adf-config.xml`:

```
<auto-purge seconds-to-live="3600"/>
```

To ensure the initial purge does not impact ongoing user activities, consider using the following WLST command to induce an MDS purge immediately before the bulk of the user load hits the system:

The following example shows how to purge all documents in the application repository whose versions are older than 10 seconds:

```
wls:/weblogic/serverConfig>purgeMetadata(application='[AppName]',server='[ServerNa
me]',olderThan=10)
```

### 31.3.7 Configuring Concurrency Management

Concurrency management includes global settings that impact the entire WebCenter Portal and service- and resource-specific settings that only impact a particular service.

You can define deployment-specific overrides or additional configuration in the `adf-config.xml` file. For example, you can specify resource-specific (producers) values that are appropriate for a particular deployment.

The following describes the format of the global, service, and resource entries in `adf-config.xml`:

```
<concurrent:adf-service-config
  xmlns="http://xmlns.oracle.com/webcenterportal/concurrent/config">
  <global
      queueSize="SIZE"
      poolCoreSize="SIZE"
      poolMaxSize="SIZE"
      poolKeepAlivePeriod="TIMEPERIOD"
      timeoutMinPeriod="TIMEPERIOD"
      timeoutMaxPeriod="TIMEPERIOD"
      timeoutDefaultPeriod="TIMEPERIOD"
      timeoutMonitorFrequency="TIMEPERIOD"
      hangMonitorFrequeny="TIMEPERIOD"
      hangAcceptableStopPeriod="TIMEPERIOD" />
  <service
      service="SERVICENAME"
      timeoutMinPeriod="TIMEPERIOD"
      timeoutMaxPeriod="TIMEPERIOD"
      timeoutDefaultPeriod="TIMEPERIOD" />
  <resource
      service="SERVICENAME"
      resource="RESOURCENAME"
      timeoutMinPeriod="TIMEPERIOD"
      timeoutMaxPeriod="TIMEPERIOD"
      timeoutDefaultPeriod="TIMEPERIOD" />
</concurrent:adf-service-config>
```

Where:

`SIZE`: A positive integer. For example: 20.

`TIMEPERIOD`: Any positive integer followed by a suffix indicating the time unit, which must be one of: `ms` for milliseconds, `s` for seconds, `m` for minutes, or `h` for hours. For example: `50ms`, `10s`, `3m`, or `1h`. The following are examples of default settings for different services. These settings are overwritten with any service-specific configurations in `connections.xml` or `adf-config.xml` files:

```
<concurrent:adf-service-config
  xmlns="http://xmlns.oracle.com/webcenter/concurrent/config">
  <service service="oracle.webcenter.community" timeoutMinPeriod="2s"
timeoutMaxPeriod="50s" timeoutDefaultPeriod="30s"/>
  <resource service="oracle.webcenter.community"
      resource="oracle.webcenter.doclib"
      timeoutMinPeriod="2s" timeoutMaxPeriod="10s" timeoutDefaultPeriod="5s"/>
  <resource service="oracle.webcenter.community"
```

```
      resource="oracle.webcenter.collab.calendar.community"
      timeoutMinPeriod="2s" timeoutMaxPeriod="10s" timeoutDefaultPeriod="5s"/>
  <resource service="oracle.webcenter.community"
      resource="oracle.webcenter.collab.rtc"
      timeoutMinPeriod="2s" timeoutMaxPeriod="10s" timeoutDefaultPeriod="5s"/>
  <resource service="oracle.webcenter.community"
      resource="oracle.webcenter.list"
      timeoutMinPeriod="2s" timeoutMaxPeriod="10s" timeoutDefaultPeriod="5s"/>
  <resource service="oracle.webcenter.community"
      resource="oracle.webcenter.collab.tasks"
      timeoutMinPeriod="2s" timeoutMaxPeriod="10s" timeoutDefaultPeriod="5s"/>
</concurrent:adf-service-config>
```

**Note:** All of the attributes except `service` and `resource` are optional, and therefore, for example, the following tags are valid:

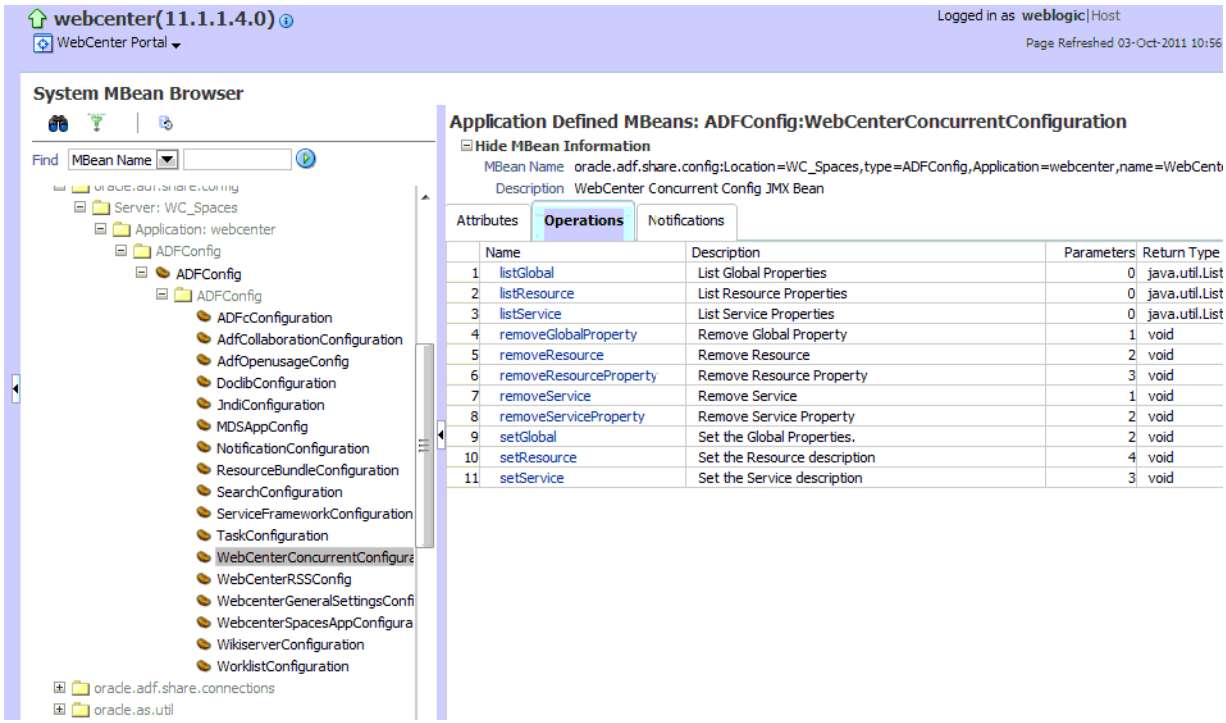
```
<global queueSize="20"/>
    <resource service="foo" resource="bar" timeoutMaxPeriod="5s"/>
```

You can use the Enterprise Manager System MBean Browser to view, add, modify, and delete the concurrency configuration based on your usage pattern. To access the MBean Browser for your WebCenter Portal application, see "Accessing the System MBean Browser" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

1. In System MBean Browser, navigate to:

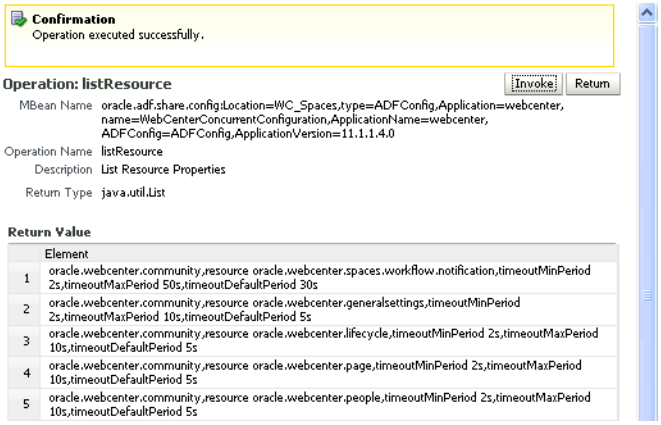
   **Application Defined MBeans** -> **oracle.adf.share.config** -> **Server: (your server name)** -> **Application: (your application name)** ->**ADFConfig** -> **ADFConfig (bean)** -> **ADFConfig** -> **WebCenterConcurrentConfiguration** -> **Operations** -> **listResource**

*Figure 31–1 System MBean Browser - WebCenterConcurrentConfiguration*



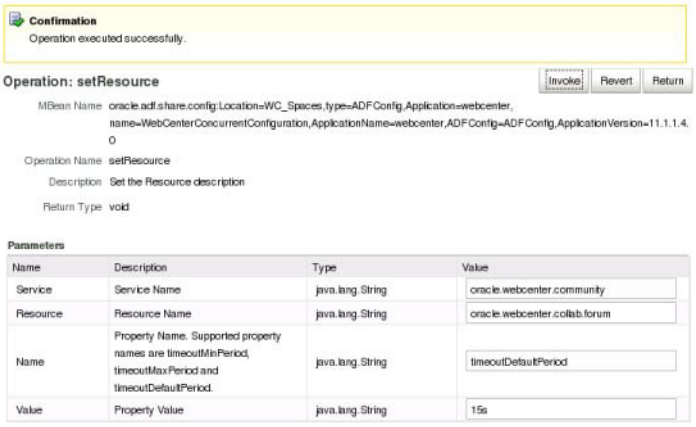

2. To view the current concurrency settings, select **listResource**, and then click **Invoke** (Figure 31–2).

*Figure 31–2 System MBean Browser - listResource*




3. To change a setting, select **setResource**, enter the resource details, and then click **Invoke** (Figure 31–3).

*Figure 31–3 System MBean Browser - setResource*




Take care to enter the correct values for **service**, **resource**, **name** and **value**.

**NOTE:** If the resource parameter you are attempting to modify already has a **[value]** setting, you must remove the setting first by invoking the **[removeResource]** operation (Figure 31–4).

*Figure 31–4 System MBean Browser - removeResource*


4. To save changes, navigate to **Application Defined MBeans: ADFConfig:ADFConfig -> save**, and click **Invoke**.

## 31.4 Tuning Back-End Component Configuration

This section describes performance configuration for back-end services used by WebCenter Portal applications. Performance of back-ends such BPEL servers or Oracle WebCenter Content servers, for example, should be tuned as described in guidelines for those back-ends.

This section includes the following sub sections:

- Tuning Performance of the Announcements Service
- Tuning Performance of the Discussions Service
- Tuning Performance of the Instant Messaging and Presence (IMP) Service

- Tuning Performance of the Mail Service
- Tuning Performance of the Personal Events Service
- Tuning Performance of the RSS News Feed Service
- Tuning Performance of the Search Service
- Tuning Policy Store Parameters

### 31.4.1 Tuning Performance of the Announcements Service

To manage overall resource usage for the Announcements service, you can tune the Connection Timeout property:

- Default: 10 seconds
- Minimum: 0 seconds
- Maximum: 45 seconds

Post deployment, modify the Connection Timeout property through Fusion Middleware Control or using WLST. For details, see:

- "Modifying Discussions Server Connection Details Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*
- "Modifying Discussions Server Connection Details Using WLST" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

The following is a sample code snippet of the connections.xml to change the default timeout to 5 seconds:

```
<Reference name="Jive-7777"
className="oracle.adf.mbean.share.connection.webcenter.Announcement.
AnnouncementConnection">
<Factory
className="oracle.adf.mbean.share.connection.webcenter.forum.ForumConnectionFactor
y"/>
          <StringRefAddr addrType="connection.time.out">
             <Contents>5</Contents>
          </StringRefAddr>
       </RefAddresses>
</Reference>
```

### 31.4.2 Tuning Performance of the Discussions Service

To manage overall resource usage for the Discussions service, you can tune the Connection Timeout property:

- Default: 10 seconds
- Minimum: 0 seconds
- Maximum: 45 seconds

Post deployment, modify the Connection Timeout property through Fusion Middleware Control or using WLST. For details, see:

- "Modifying Discussions Server Connection Details Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

- "Modifying Discussions Server Connection Details Using WLST" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

The following is a sample snippet of `connections.xml`:

```
<Reference name="Jive-7777"
className="oracle.adf.mbean.share.connection.webcenter.forum.ForumConnection">
         <Factory
className="oracle.adf.mbean.share.connection.webcenter.forum.ForumConnectionFactor
y"/>
         <RefAddresses>
            <StringRefAddr addrType="forum.url">
               <Contents>http://[machine]:[port]/owc_discussions_5520</Contents>
               <StringRefAddr addrType="connection.time.out">
               <Contents>5</Contents>
            </StringRefAddr>
         </RefAddresses>
</Reference>
```

### 31.4.3 Tuning Performance of the Instant Messaging and Presence (IMP) Service

To manage overall resource usage for the IMP service, you can tune the Connection Timeout property:

- Default: 10 seconds
- Minimum: 0 seconds
- Maximum: 45 seconds

Post deployment, modify the Connection Timeout property through Fusion Middleware Control or using WLST. For details, see:

- "Modifying Instant Messaging and Presence Connections Details Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*
- "Modifying Instant Messaging and Presence Connections Details Using WLST" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

The following is a sample code snippet of the connections.xml to change the default timeout to 5 seconds:

```
<Reference name="IMPService-LCS"
   className="oracle.adf.mbean.share.connection.webcenter.rtc.RtcConnection">
  <Factory
className="oracle.adf.mbean.share.connection.webcenter.rtc.RtcConnectionFactory"/>
   <RefAddresses>
    <StringRefAddr addrType="connection.time.out">
     <Contents>5</Contents>
    </StringRefAddr>
   </RefAddresses>
</Reference>
```

### 31.4.4 Tuning Performance of the Mail Service

To manage overall resource usage for the Mail service, you can tune the Connection Timeout property:

- Default: 10 seconds
- Minimum: 0 seconds
- Maximum: 45 seconds

Post deployment, modify the Connection Timeout property through Fusion Middleware Control or using WLST. For details, see:

- "Modifying Mail Server Connection Details Using Fusion Middleware Control" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*
- "Modifying Mail Server Connection Details Using WLST" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

The following is a sample code snippet of the connections.xml to change the default timeout to 5 seconds:

```
<Reference name="MailConnection"
className="oracle.adf.mbean.share.connection.webcenter.mail.MailConnection">
  <StringRefAddr addrType="connection.time.out">
      <Contents>5</Contents>
  </StringRefAddr>
</Reference>
```

### 31.4.5 Tuning Performance of the Personal Events Service

To manage overall resource usage for the Personal Events service, you can tune the Connection Timeout property:

- Default: 10 seconds
- Minimum: 0 seconds
- Maximum: 45 seconds

You can also set a cache expiration period:

- Default: 10 seconds
- Minimum: 0 seconds
- Maximum: 45 seconds

Post deployment, modify the Connection Timeout and Cache Expiration properties through Fusion Middleware Control or using WLST. For details, see:

- "Modifying Event Server Connection Details Using Fusion Middleware Control" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*
- "Modifying Event Server Connection Details Using WLST" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

The following is a sample code snippet of the connections.xml to change the default timeout to 5 seconds:

```
<Reference
name="MSExchange-my-pc"className="oracle.adf.mbean.share.connection.webcenter.cale
ndar.PersonalEventConnection">
  <Factory
className="oracle.adf.mbean.share.connection.webcenter.calendar.PersonalEventConne
ctionFactory"/>
  <StringRefAddr addrType="eventservice.connection.timeout">
    <Contents>5</Contents>
  </StringRefAddr>
  <StringRefAddr addrType="eventservice.cache.expiration.time">
    <Contents>5</Contents>
  </StringRefAddr>
 </RefAddresses>
</Reference>
```

### 31.4.6 Tuning Performance of the RSS News Feed Service

To manage overall resource usage for the RSS News Feed service, you can adjust the refresh interval and timeout in the `adf-config.xml` file.

If you must modify these properties, post deployment, use the System MBeans Browser.

The following is a sample snippet of `adf-config.xml`:

```
<rssC:adf-rss-config>
       <rssC:RefreshSecs>3600</rssC:RefreshSecs>
       <rssC:TimeoutSecs>3</rssC:TimeoutSecs>
       <rssC:Configured>true</rssC:Configured>
</rssC:adf-rss-config>
```

### 31.4.7 Tuning Performance of the Search Service

To manage overall resource usage and user response time for searching, you can adjust the number of saved searches displayed, the number of results displayed, and these timeout values:

- `prepareTimeoutMs` - Maximum time that a service is allowed to initialize a search (in ms).
- `timeoutMs` - Maximum time that a service is allowed to execute a search (in ms).
- `showAllTimeoutMs` - Maximum time that a service is allowed to display search all results (in ms).

Post deployment, modify timeout properties through Fusion Middleware Control or using WLST. For details, see:

- "Modifying Oracle SES Connection Details Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.
- "Modifying Oracle SES Connection Details Using WLST" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `adf-config.xml`:

```
<searchC:adf-search-config
xmlns="http://xmlns.oracle.com/webcenter/search/config">
         <display-properties>
            <common numSavedSearches="25"/>
            <region-specific>
               <usage id="simpleSearchResultUIMetadata" numServiceRows="5"/>
               <usage id="searchResultUIMetadata" numServiceRows="5"/>
               <usage id="localToolbarRegion" numServiceRows="5"/>
            </region-specific>
         </display-properties>
        <execution-properties prepareTimeoutMs="1000" timeoutMs="3000"
showAllTimeoutMs="20000" />
         </execution-properties>
</searchC:adf-search-config>
```

### 31.4.8 Tuning Policy Store Parameters

If you are experiencing performance issues post login, especially in the area of permission checks, you may need to tune the policy store parameters as described in Section 29.3.5, "OPSS PDP Service Tuning Parameters". Depending on your use case scenarios, performance of WebCenter Portal and WebCenter Portal: Spaces, specifically, can be improved by modifying the following parameters:

- Set `oracle.security.jps.policystore.rolemember.cache.warmup.enable` to True
- Modify `oracle.security.jps.policystore.rolemember.cache.size` based on the number of active groups you expect to have in your WebCenter Portal - Spaces environment.

  **NOTE:** This parameter should only be modified if you expect to have more than 3000 active Spaces in your WebCenter Portal: Spaces environment.
- Set `oracle.security.jps.policystore.policy.cache.size` to 5 times the number of group spaces

> **Note:** Always refer to your own use case scenarios before modifying the policy store parameters. For more information, see the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* before tuning any security parameters.

## 31.5 Tuning Identity Store Configuration

The following sections describe performance-related configurations that may be required for specific environments.

This section includes the following subsections:

- Section 31.5.1, "Tuning the Identity Store when Using SSL"
- Section 31.5.2, "Tuning Performance when Using OVD"
- Section 31.5.3, "Tuning Performance when Using Active Directory"

### 31.5.1 Tuning the Identity Store when Using SSL

When you configure an identity store with WebCenter Portal (using WebLogic Server providers), you can choose to configure either an SSL port or a non-SSL port. If you choose an SSL port, by default, the JNDI connections are not pooled causing increased response time and decreased performance when looking up users, groups, or other identity store entities. To address this, do the following:

1. Open the `jps-config.xml` file under `domain_home/config/fmwconfig/jps-config.xml`, locate the `idstore.ldap` service instance and add the line highlighted below:

```
<!-- JPS WLS LDAP Identity Store Service Instance -->
    <serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">
       <property name="idstore.config.provider"
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"/>
       <property name="CONNECTION_POOL_CLASS"
value="oracle.security.idm.providers.stdldap.JNDIPool"/>
       <property name="java.naming.ldap.factory.socket"
value="javax.net.ssl.SSLSocketFactory"/>
     </serviceInstance>
```

2. Restart all the servers within the domain that are connected to the identity store on an SSL port with the following JVM parameter:

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

You can specify this by modifying `setDomainEnv.sh` or directly from the console.

3. Verify that the servers are running with this JVM parameter, and then (for *nix systems) run the following grep command:

```
ps -aef | grep WC_Spaces
```

and verify that the process state specifies `com.sun.jndi.ldap.connect.pool.protocol=ssl`.

## 31.5.2 Tuning Performance when Using OVD

For OVD, the only object class against which attributes are looked up is `inetOrgPerson` (and it's parent object classes). Since the Profile Gallery can display attributes not defined in `inetOrgPerson`, all the additional attributes not covered in `inetOrgPerson` would require an additional round trip to the identity store.

For best performance when using OVD in a production environment, Oracle recommends that you add the following configuration entry (in bold) to the domain-level `jps-config.xml` file:

```
        <!-- JPS WLS LDAP Identity Store Service Instance -->
        <serviceInstance name="idstore.ldap"
         provider="idstore.ldap.provider">
            <property name="idstore.config.provider"
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"/>
            <property name="CONNECTION_POOL_CLASS"
value="oracle.security.idm.providers.stdldap.JNDIPool"/>

          <extendedProperty>
            <name>user.object.classes</name>
            <values>
               <value>top</value>
               <value>person</value>
               <value>inetorgperson</value>
               <value>organizationalperson</value>
               <value>orcluser</value>
               <value>orcluserv2</value>
               <value>ctCalUser</value>
            </values>
          </extendedProperty>
        </serviceInstance>
```

## 31.5.3 Tuning Performance when Using Active Directory

For best performance when using Active Directory in a production environment, Oracle recommends that you add the following configuration entries (in bold) to the domain-level `jps-config.xml` file:

```
        <serviceInstance provider="idstore.ldap.provider"
         name="idstore.ldap">
           <property
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvider"
            name="idstore.config.provider"/>
          <property value="oracle.security.idm.providers.stdldap.JNDIPool"
           name="CONNECTION_POOL_CLASS"/>
          <property name="PROPERTY_ATTRIBUTE_MAPPING" value="WIRELESS_ACCT_
NUMBER=mobile:MIDDLE_NAME=middlename:MAIDEN_NAME=sn:DATE_OF_HIRE=pwdLastSet:NAME_
SUFFIX=generationqualifier:DATE_OF_BIRTH=pwdLastSet:DEFAULT_GROUP=primaryGroupID"
```

```
/>
          <property value="sAMAccountName" name="username.attr"/>
          <property value="sAMAccountName" name="user.login.attr"/>
        </serviceInstance>
```

The People Profile Service queries for all these attributes and there is no default mapping for these attributes in the Active Directory provider. A vanilla Active Directory installation doesn't have any mapping corresponding to `DATE_OF_HIRE`, `DATE_OF_BIRTH`.

Note that the two attributes are simply a mapping to some attribute of the correct data type to reduce unnecessary LDAP server calls as Active Directory really doesn't have corresponding attributes with the same semantic meaning.

## 31.6 Tuning Portlet Configuration

This section describes portlet performance-related configuration. This section includes the following sub sections:

- Tuning Performance of the Portlet Service
- Configuring Portlet Cache Size
- Enabling Java Object Cache for WSRP Producers
- Suppressing Optimistic Rendering for WSRP Portlets
- Tuning Performance of Oracle PDK-Java Producers
- Setting Portlet Container Runtime Options
- Setting DefaultServedResourceRequiresWsrpRewrite for WSRP Portlets
- Setting DefaultProxiedResourceRequiresWsrpRewrite for WSRP Portlets
- Importing Consumer CSS Files in IFrame Portlets
- Configuring Portlet Timeout
- Tuning Performance of OmniPortlet

### 31.6.1 Tuning Performance of the Portlet Service

To manage overall resource usage and user response time, you can remove unnecessary locale support, modify portlet timeout and cache size in the `adf-config.xml` file.

For the Portlet service, 28 supported locales are defined out-of-the-box. You can remove the locales that are unnecessary for your application.

If you must modify these properties, post deployment, you must edit `adf-config.xml` manually. See "Editing adf-config.xml" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `adf-config.xml`:

```
<portletC:adf-portlet-config xmlns="http://xmlns.oracle.com/adf/portlet/config">
        <supportedLocales>
          <value>es</value>
          <value>ko</value>
          <value>ru</value>
          <value>ar</value>
          <value>fi</value>
          <value>nl</value>
```

```
            <value>sk</value>
            <value>cs</value>
            <value>fr</value>
            <value>no</value>
            <value>sv</value>
            <value>da</value>
            <value>hu</value>
            <value>pl</value>
            <value>th</value>
            <value>de</value>
            <value>it</value>
            <value>pt</value>
            <value>tr</value>
            <value>el</value>
            <value>iw</value>
            <value>pt_BR</value>
            <value>zh_CN</value>
            <value>en</value>
            <value>ja</value>
            <value>ro</value>
            <value>zh_TW</value>
          </supportedLocales>
          <defaultTimeout>20</defaultTimeout>
          <minimumTimeout>1</minimumTimeout>
          <maximumTimeout>300</maximumTimeout>
          <parallelPoolSize>10</parallelPoolSize>
          <parallelQueueSize>20</parallelQueueSize>
          <cacheSettings enabled="true">
            <maxSize>10000000</maxSize>
          </cacheSettings>
</portletC:adf-portlet-config>
```

## 31.6.2 Configuring Portlet Cache Size

You can modify the portlet cache size in the `adf-config.xml` file. The default portlet cache size is set to 10 MB.

If you must modify these properties, post deployment, you must edit `adf-config.xml` manually.

The following is a sample snippet of `adf-config.xml`:

```
<adf-portlet-config>
     ....
      <supportedLocales>
      <cacheSettings enabled="true">
    <maxSize>10000000</maxSize>
      </cacheSettings>
</adf-portlet-config>
```

## 31.6.3 Enabling Java Object Cache for WSRP Producers

Oracle recommends that you enable the Java Object Cache (JOC) for WSRP producers so that objects written to the persistent store are cached.

The following is a sample snippet of `web.xml`:

```
<env-entry>
  <env-entry-name>oracle/portal/wsrp/server/enableJavaObjectCache
  </env-entry-name>
  <env-entry-type>java.lang.String
```

```
  </env-entry-type>
  <env-entry-value>false
  </env-entry-value>
</env-entry>
```

### 31.6.4 Suppressing Optimistic Rendering for WSRP Portlets

To suppress the optimistic render of WSRP portlets after a WSRP `PerformBlockingInteraction` or `HandleEvents` call, set the Portlet container runtime option (specified in portlet.xml) as follows: `com.oracle.portlet.suppressWsrpOptimisticRender=true`.

- `true` - optimistic render always suppressed
- `false` - optimistic render may be performed

Normally, if a WSRP portlet receives a WSRP `PerformBlockingInteraction` request (`processAction` in JSR168/JSR286 portlets) and the portlet does not send any events as a result, the WSRP producer renders the portlet and returns the portlet's markup in the response to the `PerformBlockingInteraction` SOAP message. This markup may be cached by the consumer until the consumer's page renders, and if nothing else affecting the state of the portlet happens (such as the portlet receiving an event), the cached markup can be used by the consumer, eliminating the need for a second SOAP call to `GetMarkup`.

This assumes that the portlet's render phase is idempotent, which is always a best practice. However, if the portlet expects to receive an event, or rendering the portlet is more costly than a second SOAP message for `GetMarkup`, the developer may use this container option to suppress the optimistic render of the portlet after a `processAction` or `handleEvent` call. The portlet still renders normally when the producer receives the WSRP `GetMarkup` request.

### 31.6.5 Tuning Performance of Oracle PDK-Java Producers

To manage overall resource usage for a Web producer, you can tune the Connection Timeout property:

- Default: 30000 ms
- Minimum: 5000 ms
- Maximum: 60000 ms

Post deployment, modify the Connection Timeout property through Fusion Middleware Control or using WLST. For details, see:

- "Editing Producer Registration Details Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.
- "Editing Producer Registration Details Using WLST" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `connections.xml`:

```
<webproducerconnection producerName="wc-WebClipping"
urlConnection="wc-WebClipping-urlconn" timeout="10000" establishSession="true"
mapUser="false"/>
```

### 31.6.6 Setting Portlet Container Runtime Options

You can use the WebCenter Portal-specific `excludedActionScopeRequestAttributes` container runtime option to specify

how to store action-scoped request attributes so that they are available to portlets until a new action occurs.

Request attributes which match any of the regular expressions are not stored as action-scoped request attributes if the `javax.portlet.actionScopedRequestAttributes` container runtime option is used, in addition to any request parameters whose values match the regular expressions defined in the com.oracle.portlet.externalScopeRequestAttributes container runtime option.

If set to true, you can specify a second value of `numberOfCachedScopes` and a third value indicating the number of scopes to be cached by the portlet container.

## 31.6.7 Setting DefaultServedResourceRequiresWsrpRewrite for WSRP Portlets

To specify the default WSRP `requiresRewrite` flag to use when generating Resource URLs for portlet-served resources, set the Portlet container runtime option (specified in portlet.xml) as follows: `com.oracle.portlet.defaultServedResourceRequiresWsrpRewrite`.

This setting is used for all ResourceURLs created by the portlet, unless overridden by the presence of the `oracle.portlet.server.resourceRequiresRewriting` request attribute when the ResourceURL methods `write()` or `toString()` are called. This setting is also used to specify the WSRP `requiresRewriting` flag on the served resource response, but can be overridden by the presence of the `oracle.portlet.server.resourceRequiresRewriting` request attribute when the portlet's `serveResource()` method returns.

Valid values:

- `unspecified` - (Default) The `requiresRewrite` URL flag is not given a value, and the `requiresRewriting` response flag for a `serveResource` operation is based on the MIME type of the response.
- `true` - The `requiresRewrite` URL flag and `requiresRewriting` response flag is set to `true`, indicating that the resource should be rewritten by the consumer.
- `false` - The `requiresRewrite` URL flag and `requiresRewriting` response flag is set to `false`, indicating that the resource does not necessarily need to be rewritten by the consumer, though the consumer may choose to rewrite the resource.

## 31.6.8 Setting DefaultProxiedResourceRequiresWsrpRewrite for WSRP Portlets

To specify the default WSRP `requiresRewrite` flag to use when encoding URLs for resources not served by the portlet, set the Portlet container runtime option (specified in portlet.xml) as follows: `com.oracle.portlet.defaultProxiedResourceRequiresWsrpRewrite`.

This setting is used for all URLs returned by the `PortletResponse.encodeURL()` method, unless overridden by the presence of the `oracle.portlet.server.resourceRequiresRewriting` request attribute when the `PortletResponse.encodeURL()` method is called.

Valid values:

- `true` - (Default) The `requiresRewrite` URL flag is set to true, indicating that the resource should be rewritten by the consumer.

- `false` - The `requiresRewrite` URL flag is set to false, indicating that the resource does not necessarily need to be rewritten by the consumer.

### 31.6.9 Importing Consumer CSS Files in IFrame Portlets

To specify to a portal consumer that the CSS file is imported to an IFramed portlet, set the Portlet container runtime option (specified in portlet.xml) as follows: `com.oracle.portlet.importCssToIFrame`.

Valid values:

- `true` - The CSS file from the consumer is applied to an IFramed portlet.
- `false` - (Default) Nothing is done.

### 31.6.10 Configuring Portlet Timeout

You can modify the portlet timeout value in the `adf-portlet-config` element of the `adf-config.xml` file. Default: 10 seconds, minimum: 0.1 seconds, maximum: 60 seconds.

If you must modify these properties, post deployment, you must edit `adf-config.xml` manually. See "Editing adf-config.xml" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `adf-config.xml`:

```
<adf-portlet-config>
     ....
  <defaultTimeout>5</defaultTimeout>
  <minimumTimeout>2</minimumTimeout>
  <maximumTimeout>300</maximumTimeout>
</adf-portlet-config>
```

### 31.6.11 Tuning Performance of OmniPortlet

To manage overall resource usage for OmniPortlets, you can tune the Connection Timeout property:

- Default: 30000 ms
- Minimum: 5000 ms
- Maximum: 60000 ms

Post deployment, modify the Connection Timeout property through Fusion Middleware Control or using WLST. For details, see:

- "Editing Producer Registration Details Using Fusion Middleware Control" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.
- "Editing Producer Registration Details Using WLST" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

The following is a sample snippet of `connections.xml`:

```
<webproducerconnection producerName="wc-OmniPortlet"
urlConnection="wc-OmniPortlet-urlconn" timeout="10000" establishSession="false"
mapUser="false"/>
```

# Part VII

## Capacity Planning, Scalability, and Availability

This part describes how to plan your site for high traffic, scalability, and availability. It contains the following chapters:

# 32

# Capacity Planning

Capacity Planning is the process of determining what type of hardware and software configuration is required to meet application needs. Like performance planning, capacity planning is an iterative process. A good capacity management plan is based on monitoring and measuring load data over time and implementing flexible solutions to handle variances without impacting performance.

> **Note:** The information contained in this chapter is meant to provide an overview of various techniques that can be used to develop an effective capacity management plan. The steps you take - and the plan you ultimately create - depends on your specific requirements and deployment structure.

The following sections provide an introduction to capacity planning:

- Section 32.1, "About Capacity Planning for Oracle Fusion Middleware"
- Section 32.2, "Determining Performance Goals and Objectives"
- Section 32.3, "Measuring Your Performance Metrics"
- Section 32.4, "Identifying Bottlenecks in Your System"
- Section 32.5, "Implementing a Capacity Management Plan"

## 32.1 About Capacity Planning for Oracle Fusion Middleware

While performance tuning can be defined as optimizing your *existing* system for better performance, capacity planning determines what your system needs (and when it needs it) to maintain performance in both steady-state and peak usage periods.

Capacity Planning involves designing your solution and testing the configuration, as well as identifying business expectations, periodic fluctuations in demand, and application constraints. You need to plan carefully, test methodically, and incorporate design principles that focus on performance. Before deploying any application into a production environment, the application should be put through a rigorous performance testing cycle. Creating an effective Capacity Management plan includes some of the same steps as performance planning:

- Step 1:Determining Performance Goals and Objectives
- Step 2: Measuring Your Performance Metrics
- Step 3: Identifying Bottlenecks in Your System
- Step 4: Implementing a Capacity Management Plan

### 32.1.1 Capacity Planning Factors to Consider

Before you can create a plan, you must have the data to support your deployment strategy. The following list of questions should be asked - and the information you receive should be analyzed carefully - to ensure a successful capacity management plan.

***Table 32–1 Capacity Planning Factors to Consider***

| Capacity Planning Questions | For more information see, |
|---|---|
| What are your performance goals and objectives? | Section 32.2, "Determining Performance Goals and Objectives" |
| How many users need to run simultaneously (concurrently?) | Section 32.2, "Determining Performance Goals and Objectives" |
| Is the simulated workload adequate? (Is the workload likely to increase?) | Section 32.2, "Determining Performance Goals and Objectives" |
| Is the Oracle Fusion Middleware deployment configured to support clustering and other high availability factors? | Section 32.4.1, "Using Clustered Configurations" |
| Does the hardware meet the configuration requirements? | Section 32.5.1, "Hardware Configuration Requirements" |
| Do you have adequate JVMs to support your users? | Section 32.5.2, "JVM Requirements" |
| Is the database a limiting factor? | Section 32.5.4, "Database Configuration" |

## 32.2 Determining Performance Goals and Objectives

The first step in creating an effective capacity management plan is to determine your network load and performance objectives. You need to understand the applications deployed and the environmental constraints placed on the system. Ideally you have information about the levels of activity that components of the application are expected to meet, such as:

- The anticipated number of users.
- The number of concurrent sessions.
- The number of SSL connections required.
- The number and size of requests.
- The amount of data and its consistency.
- Determining your target CPU utilization.

Performance objectives are limited by constraints, such as

- The configuration of hardware and software such as CPU type, disk size versus disk speed, sufficient memory.
- The ability to interoperate between domains, use legacy systems, support legacy data.
- The security requirements and use of SSL. SSL involves intensive computing operations and supporting the cryptography operations in the SSL protocol can impact the performance of the WebLogic Server.
- Development, implementation, and maintenance costs.

You can use this information to set realistic performance objectives for your application environment, such as response times, throughput, and load on specific hardware.

## 32.3 Measuring Your Performance Metrics

After you have determined your performance criteria in Section 32.2, "Determining Performance Goals and Objectives", take measurements of the metrics you can use to quantify your performance objectives. Benchmarking key performance indicators provides a performance baseline. See Chapter 4, "Monitoring Oracle Fusion Middleware" for information on measuring your performance metrics with Oracle Fusion Middleware applications.

## 32.4 Identifying Bottlenecks in Your System

Bottlenecks, or areas of marked performance degradation, should be addressed while developing your capacity management plan. If possible, profile your applications to pinpoint bottlenecks and improve application performance. Oracle provides the following profilers:

- Oracle Jrockit Mission Control provides profiling capabilities for processes using Jrockit JVM.

  `http://www.oracle.com/technology/products/jrockit/missioncontrol/index.html`

- Oracle Application Diagnostics provides profiling capabilities for java processing using SUN JDK.

  `http://www.oracle.com/technology/software/products/oem/htdocs/jade.html`

The objective of identifying bottlenecks is to meet your performance goals, not eliminate all bottlenecks. Resources within a system are finite. By definition, at least one resource (CPU, memory, or I/O) can be a bottleneck in the system. Planning for anticipated peak usage, for example, may help minimize the impact of bottlenecks on your performance objectives.

There are several ways to address system bottlenecks. Some common solutions include:

- Using Clustered Configurations
- Using Connection Pooling
- Setting the Max Heap Size on JVM
- Increasing Memory or CPU
- Segregation of Network Traffic
- Segregation of Processes and Hardware Interrupt Handlers

### 32.4.1 Using Clustered Configurations

Clustered configurations distribute work loads among multiple identical cluster member instances. This effectively multiplies the amount of resources available to the distributed process, and provides for seamless fail over for high availability.

For more information see Chapter 33, "Using Clusters and High Availability Features".

### 32.4.2 Using Connection Pooling

You may be able to improve performance by using existing database connections. You can limit the number of connections, timing of the sessions and other parameters by modifying the connection strings.

See Section 2.7, "Reusing Database Connections" for more information on configuring the database connection pools.

### 32.4.3 Setting the Max Heap Size on JVM

This is a application-specific tunable that enables a trade off between garbage collection times and the number of JVMs that can be run on the same hardware. Large heaps are used more efficiently and often result in fewer garbage collections. More JVM processes offer more fail over points.

See Section 2.4, "Tuning Java Virtual Machines (JVMs)" for more information.

### 32.4.4 Increasing Memory or CPU

Aggregating more memory and/or CPU on a single hardware resource allows localized communication between the instances sharing the same hardware. More physical memory and processing power on a single machine enables the JVMs to scale and run much larger and more powerful instances, especially 64-bit JVMs. Large JVMs tend to use the memory more efficiently, and Garbage Collections tend to occur less frequently. In some cases, adding more CPU means that the machine can have more instruction and data cache available to the processing units, which means even higher processing efficiency.

See Section 2.2, "Securing Sufficient Hardware Resources" for more information.

### 32.4.5 Segregation of Network Traffic

Network-intensive applications can introduce significant performance issues for other applications using network. Segregating the network traffic of time-critical applications from network-intensive applications, so that they get routed to different network interfaces, may reduce performance impacts. It is also possible to assign different routing priorities to the traffic originating from different network interfaces.

### 32.4.6 Segregation of Processes and Hardware Interrupt Handlers

When planning for the capacity that a specific hardware resource can handle, it is important to understand that the operating system may not be able to efficiently schedule the JVM processes as well as other system processes and hardware interrupt handlers. The JVM may experience performance impacts if it shares even a few of its CPU cores with the hardware interrupt handlers. For example, disk and network-intensive applications may induce performance impacts that are disproportionate to the load experienced by the CPU. In addition, hardware interrupts can prevent the active Java threads from reaching a "GC-safe point" efficiently. Separating frequent hardware interrupt handlers from the CPUs running the JVM process can reduce the wait for Garbage Collections to start.

It may also be beneficial to dedicate sibling CPUs on a multi-core machine to a single JVM to increase the efficiency of its CPU cache. If multiple processes have to share the CPU, the data and instruction cache can be contaminated with the data and instructions from both processes, thus reducing the amount of the cache used effectively. Assigning the processes to specific CPU cores, however, can make it impossible to use other CPU cores during peak load bursts. The capacity management

plan should include a determination on whether the CPUs should be used more efficiently for the nominal load, or should there be some extra capacity for a burst of activity.

## 32.5 Implementing a Capacity Management Plan

Once you have defined your performance objectives, measured your workload, and identified any bottlenecks, you must create and implement a capacity management plan. The goal of your plan should be to meet or exceed your performance objectives (especially during peak usage periods) and to allow for future workload increases. To achieve your performance objectives, you must implement your management plan and then continuously monitor the performance metrics as discussed in Chapter 4, "Monitoring Oracle Fusion Middleware".

Since no two deployments are identical, its virtually impossible to illustrate how a capacity management plan would be implemented for all configurations. Capacity planning is an iterative process and your plan must be calibrated as changes in your workload or environment change. The following section provides key factors that should be addressed in the plan:

### 32.5.1 Hardware Configuration Requirements

There is no single formula for determining your hardware requirements. The process of determining what type of hardware and software configuration involves assessment of your system performance goals and an understanding of your application. Capacity planning for server hardware should focus on maximum performance requirements.

The hardware requirements you have today are likely to change. Your plan should allow for workload increases, environment changes (such as added servers or 3rd party services), software upgrades (operating systems, middleware or other applications), network connectivity and network protocols.

#### 32.5.1.1 CPU Requirements

Your target CPU usage should not be 100%, you should determine a target CPU utilization based on your application needs, including CPU cycles for peak usage. If your CPU utilization is optimized at 100% during normal load hours, you have no capacity to handle a peak load. In applications that are latency sensitive and maintaining the ability for a fast response time is important, high CPU usage (approaching 100% utilization) can reduce response times while throughput stays constant or even increases because of work queuing up in the server. For such applications, a 70% - 80% CPU utilization recommended. A good target for non-latency sensitive applications is about 90%.

#### 32.5.1.2 Memory Requirements

Memory requirements are determined by the optimal heap size for the applications you are going to use, for each JVM co-located on the same hardware. Each JVM needs up to 500MB in addition to the optimal heap size; the actual impact to performance depends on the JVM brand, and on the type of application being run. For example, applications with more Java classes loaded need more space for compiled classes. 32-bit JVMs normally cannot exceed a limit of approximately 3GB on some architecture when a limit is imposed by the hardware architecture and the Operating System. It is recommended to reserve some memory for the Operating System, IO buffers and shared-memory devices.

### 32.5.2 JVM Requirements

The number of users/processes that a single Java Virtual Machine (JVM) can handle varies widely on the types of requests and the type of JVM you are running. As part of your performance monitoring and benchmarking procedures, you should determine how many and what kinds of processes are executed and determine if your hardware meets the requirements for your specific JVM.

### 32.5.3 Managed Servers

Using multiple managed servers across multiple nodes in a clustered configuration is recommended for both high performance and reliability. It is important to note, however, that having multiple managed servers may mean using more memory which can enable some applications to optimize certain operations in-memory, therefore reducing impact of disk, database and network latency.

For more information on using clustered configurations, see "Understanding Managed Servers and Managed Server Clusters" in *Oracle Fusion Middleware Administrator's Guide*.

### 32.5.4 Database Configuration

To maintain sustained performance, you must ensure that your existing database can scale with the increases in capacity planned for the application server tier. Tuning the database parameters and monitoring database metrics during peak usage, can help you determine if the existing database resources can scale to handle increased loads. You may need to add additional memory or upgrade the database hardware configuration. For more information on tuning an Oracle database, see the *Oracle Database Performance Tuning Guide*.

In some cases, however, you may find that the database is still not able to effectively manage increases in load, even after increasing the memory or upgrading the CPU. In these situations, consider deploying an Oracle Real Application Cluster (Oracle RAC) environment to handle the increases. Oracle RAC configurations not only provide enhanced performance, but they can also improve reliability and scalability. For more information on Oracle RAC, see *Oracle Real Application Clusters Administration and Deployment Guide*.

# 33

# Using Clusters and High Availability Features

A high availability architecture is one of the key requirements for any Enterprise Deployment. Oracle Fusion Middleware has an extensive set of high availability features, which protect its components and applications from unplanned down time and minimize planned downtime.

This chapter provides an overview of the architecture, interaction, and dependencies of Oracle Fusion Middleware components, and explains how they can be deployed in a high availability architecture to maximize performance.

This chapter includes the following sections:



> **Note:** Using clusters and other high availability options is a complex and detailed process. This chapter is meant to introduce the concepts as they relate to Oracle Fusion Middleware. Table 33–1 provides a list of Oracle Fusion Middleware guides that contain detailed high availability information.

## 33.1 About Clusters and High Availability Features

One of the most important factors in both high availability and performance is the use of **clusters**. A cluster is a set of processes running on single or multiple computers that share the same workload. Using a clustered configuration promotes scalability, high availability, and performance.

**High availability** refers to the ability of users to access a system without loss of service. Deploying a high availability system minimizes the time when the system is down, or unavailable and maximizes the time when it is running, or available. See

Details about using clusters and other high availability features can be located in the application-specific guides listed in Table 33–1:

***Table 33–1 Clusters and High Availability Information in Oracle Fusion Middleware Documentation***

| Component | Location of Information |
|---|---|
| Oracle Fusion Middleware | *Oracle Fusion Middleware Administrator's Guide* |
| Oracle WebLogic Server | *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*<br>*Oracle Fusion Middleware Performance and Tuning for Oracle WebLogic Server* |
| Oracle SOA Suite | The *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*<br>The *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite* |
| Oracle WebCenter | The *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*<br>The *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle WebCenter Portal* |
| Oracle ADF | The *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*<br>The *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* |
| Oracle Fusion Middleware Backup and Recovery | The *Oracle Fusion Middleware Administrator's Guide* |
| Oracle Web Cache | The *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache* |
| Oracle Identity Management | The *Oracle Fusion Middleware Installation Guide for Oracle Identity Management*<br>The *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management* |
| Oracle Virtual Directory | The *Oracle Fusion Middleware Administrator's Guide for Oracle Virtual Directory* |
| Oracle HTTP Server | The *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server* |
| Oracle Internet Directory | The *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* |
| Oracle Repository Creation Utility (RCU) | The *Oracle Fusion Middleware Repository Creation Utility User's Guide* |
| Oracle Portal | The *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* |

## 33.2 Using Clusters with Oracle Fusion Middleware

For production environments that require increased application performance, throughput, or high availability, you can configure two or more Managed Servers to operate as a cluster. A cluster is a collection of multiple Oracle WebLogic Server server instances running simultaneously and working together to provide increased scalability and reliability.

For more information on using clusters with Oracle Fusion Middleware, see the following:

- "Understanding Managed Servers and Managed Server Clusters" in *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*
- *Oracle Real Application Clusters Administration and Deployment Guide*

## 33.3 Using High Availability Features with Oracle Fusion Middleware

In addition to using a clustered architecture within your Fusion Middleware components, there are a number of high availability features built-in to ensure your applications are continuously accessible by the users. The following list provides a few options for setting up a comprehensive high availability system. The options that you integrate depend on your overall performance goals as well as your system architecture. This list is meant to provide examples only.

- **Process death detection and automatic restart**

  Processes may die unexpectedly due to configuration or software problems. A proper process monitoring and restart system should constantly check the health of the applications and restart them when problems appear.

  A system process should also maintain the number of restarts within a specified time interval. This is also important since continually restarting within short time periods may lead to additional faults or failures. Therefore a maximum number of restarts or retries within a specified time interval should also be designed as well.

- **State replication and routing**

  For stateful applications, client state can be replicated to enable stateful failover of requests in the event that processes servicing these requests fail.

- **Failover**

  With a load-balancing mechanism in place, the instances are redundant. If any of the instances fail, requests to the failed instance can be sent to the surviving instances.

- **Server load balancing**

  When multiple instances of identical server components are available, client requests to these components can be load balanced to ensure that the instances have roughly the same workload.

- **Disaster Recovery**

  Disaster recovery solutions typically set up two homogeneous sites, one active and one passive. Each site is a self-contained system. The active site is generally called the production site, and the passive site is called the standby site. During normal operation, the production site services requests; in the event of a site failover or switchover, the standby site takes over the production role and all requests are routed to that site. To maintain the standby site for failover, not only must the standby site contain homogeneous installations and applications, data and configurations must also be synchronized constantly from the production site to the standby site.

For more information see the *Oracle Fusion Middleware High Availability Guide*.