# Oracle® Fusion Middleware

# Tuning Performance Guide

12*c* (12.2.1.2.0)
E77879-03
January 2018

Oracle Fusion Middleware Tuning Performance Guide, 12*c* (12.2.1.2.0)

E77879-03



Primary Author: Sreevalli Setty

# Contents

## 3 Performance Planning



## 4 Monitoring



## 5 Using the Oracle Dynamic Monitoring Service

## 7 Tuning Oracle Metadata Service



## 8 Tuning Oracle Fusion Middleware Security

# 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.



## Audience

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

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

## Conventions

The following text conventions are used in this document:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *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

Performance tuning is essential for improving system performance. Therefore, it is important to understand the basic performance concepts and how to design applications for performance and scalability.

This part contains the following topics:

- Introduction and Roadmap
  *Oracle Fusion Middleware Tuning Performance* addresses topics pertinent to application developers, Oracle Fusion Middleware administrators, database administrators, and web masters.
- Top Performance Areas
  By identifying top performance areas, you can tune Oracle Fusion Middleware for optimal performance.
- Performance Planning
  A clearly defined plan for achieving your performance objectives is essential for deciding what to trade for higher performance.
- Monitoring
  Oracle Fusion Middleware provides a variety of technologies and tools that monitor server and application performance.
- Using the Oracle Dynamic Monitoring Service
  The Oracle Dynamic Monitoring Service (DMS) publishes component performance data.

# 1
# Introduction and Roadmap

*Oracle Fusion Middleware Tuning Performance* addresses topics pertinent to application developers, Oracle Fusion Middleware administrators, database administrators, and web masters.

- Document Scope and Audience
  *Oracle Fusion Middleware Tuning Performance* is intended for application developers, Oracle Fusion Middleware administrators, database administrators, and web masters.
- Guide to this Document
  The *Oracle Fusion Middleware Tuning Performance* guide contains the following topics.
- Related Documentation
  For more information, see the following documents in the Oracle Fusion Middleware 12*c* (12.2.1.2) documentation set:

## 1.1 Document Scope and Audience

*Oracle Fusion Middleware Tuning Performance* is intended for 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

The *Oracle Fusion Middleware Tuning Performance* guide contains the following topics.

- Top Performance Areas describes top tuning areas for Oracle Fusion Middleware and serves as a Quick Start for tuning applications.
- Performance Planning describes the performance planning methodology and tuning concepts for Oracle Fusion Middleware.
- Monitoring 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.
- Using the Oracle Dynamic Monitoring Service provides an overview and features available in the Oracle Dynamic Monitoring Service (DMS).
- Tuning Oracle HTTP Server discusses the techniques for optimizing Oracle HTTP Server performance and 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.

- Tuning Oracle Metadata Service provides tuning tips for Oracle Metadata Service (MDS). MDS is used by Oracle Application Development Framework to manage metadata.
- Tuning Oracle Fusion Middleware Security describes tuning Oracle Platform Security for Java for optimal performance. 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.
- Tuning Oracle Application Development Framework (ADF) provides basic guidelines on how to maximize the performance and scalability of the ADF stack in applications. Oracle ADF is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications.
- Tuning Oracle TopLink provides some of the available performance options for Java Persistence API (JPA) entity architecture. Oracle TopLink includes EclipseLink as the JPA implementation.
- Tuning the SOA Infrastructure describes the common SOA infrastructure tuning parameters for configuring Oracle Service-Oriented Architecture (SOA) Suite components to improve performance. Oracle SOA Suite 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.
- Tuning Oracle BPEL Process Manager provides several BPEL property settings that can be configured to optimize performance at the process, domain, and application server levels.
- Tuning Oracle Mediator describes how to tune Oracle Mediator, a service engine within the Oracle SOA Service Infrastructure, for optimal performance. The Mediator service engine runs with the SOA Service Infrastructure Java EE application.
- Tuning Oracle Managed File Transfer describes how to tune Oracle Managed File Transfer, a new product in 12*c* (12.2.1). Oracle Managed File Transfer (MFT) is a high performance, standards-based, end-to-end managed file gateway.
- Tuning Oracle Business Rules describes the technology that enables automation of business rules; it also discusses the extraction of business rules from procedural logic such as Java code or BPEL processes.
- Tuning Oracle Business Process Management describes how to tune Oracle Business Process Management (BPM), which provides a seamless integration of all stages of the application development life cycle from design-time and implementation to run-time and application management.
- Tuning Oracle Human Workflow describes how to tune Oracle Human Workflow for optimal performance. Oracle Human Workflow is a service engine running in Oracle SOA Service Infrastructure that allows the execution of interactive human driven processes. The Human Workflow service consists of a number of services that handle various aspects of human interaction with a business process.
- Tuning Oracle Business Activity Monitoring describes how to tune the Oracle Business Activity Monitoring dashboard application for optimal performance. Oracle Business Activity Monitoring (BAM) provides the tools for monitoring business services and processes in the enterprise.

- Tuning Oracle Service Bus describes how to tune the Oracle Service Bus, which provides connectivity, routing, mediation, management and also some process orchestration capabilities. It is part of a larger system where it plays the role of an intermediary between two or more applications (servers).
- Tuning Oracle Enterprise Scheduler Service describes how to tune the Oracle Enterprise Scheduler Service, which enables scheduling and running jobs within a particular time frame, or workshift, using rules to create work assignments.
- Tuning Oracle Business Intelligence Performance describes how to tune Oracle Business Performance, which 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.
- Tuning Oracle WebCenter Portal describes how to tune Oracle WebCenter Portal, which 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).

## 1.3 Related Documentation

For more information, see the following documents in the Oracle Fusion Middleware 12*c* (12.2.1.2) documentation set:

- *Oracle Fusion Middleware Understanding Oracle Fusion Middleware*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*
- *Oracle Fusion Middleware Administering Oracle HTTP Server*

# 2
# Top Performance Areas

By identifying top performance areas, you can tune Oracle Fusion Middleware for optimal performance.

- Identifying Top Performance Areas
  One of the most challenging aspects of performance tuning is knowing where to begin. Therefore, it is important to identify the top performance areas for Oracle Fusion Middleware.
- Securing Sufficient Hardware Resources
  Manage the performance of Oracle Fusion Middleware applications to ensure tthere is sufficient CPU, memory, and network resources to support the user and application requirements for installation.
- Tuning the Operating System
  Each operating system has native tools and utilities that can be useful for monitoring and tuning purposes.
- Tuning Java Virtual Machines (JVMs)
  How you tune your Java virtual machine (JVM) greatly affects the performance of Oracle Fusion Middleware and your applications.
- Tuning the WebLogic Server
  Tune the WebLogic Server to match your application needs.
- 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.
- Reusing Database Connections
  It is important to tune the connection pool attributes in the JDBC data sources in your WebLogic Server domain correctly to improve application and system performance.
- Enabling Data Source Statement Caching
  Statement caching improves performance by caching executable statements that are used repeatedly.
- Controlling Concurrency
  Limiting concurrency at multiple layers of the system to match specific usage needs can greatly improve performance.
- Setting Logging Levels
  The amount of information that is logged can have a significant impact on the performance.

# 2.1 Identifying Top Performance Areas

One of the most challenging aspects of performance tuning is knowing where to begin. Therefore, it is important to identify the top performance areas for Oracle Fusion Middleware.

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 a 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 .

**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 resource requirements to maximize performance.<br>See 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 Tuning the Operating System. |
| Java Virtual Machines (JVMs) | Follow the best practices and practical tips to tune the JVM. It also helps improve the performance of a Java EE application, including heap size and JVM garbage collection options.<br>See Tuning Java Virtual Machines (JVMs). |
| Database | For applications that access a database, ensure that your database is properly configured to support requirements of the application.<br>See Tuning Database Parameters. |
| WebLogic Server | If your Oracle Fusion Middleware applications are using WebLogic Server, see Tuning the WebLogic Server. |
| Database Connections | Pooling the connections so they are reused is an important tuning consideration.<br>See 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 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 Controlling Concurrency. |
| Concurrency | Control concurrency with Oracle Fusion Middleware components.<br>See Controlling Concurrency. |

**Table 2-1 (Cont.) Top Performance Areas for Oracle Fusion Middleware**

| Performance Area | Description and Reference |
|---|---|
| Logging Levels | Logging levels are thresholds that a system administrator sets to control how much information is logged. Set the logging levels appropriately as it impacts the performance by the amount of information that applications log.<br>See Setting Logging Levels. |

# 2.2 Securing Sufficient Hardware Resources

Manage the performance of Oracle Fusion Middleware applications to ensure tthere is sufficient CPU, memory, and network resources to support the user and application requirements for installation.

No matter how well you tune your applications, if the appropriate hardware resources are not used, your applications cannot reach their optimal performance levels. Oracle Fusion Middleware has minimum hardware requirements for its applications and database tier. For details on Oracle Fusion Middleware supported configurations, see Verifying Certification, System Requirements, and Interoperability in *Oracle Fusion Middleware Planning an Installation of Oracle Fusion Middleware*.

Sufficient hardware resources must 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 details on monitoring, see Monitoring .

> **Tip:**
>
> Your target CPU usage should never reach 100% utilization. Target the 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, maintaining a fast response time is important. High CPU usage (approaching 100% utilization) can increase response time 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 might 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 any traffic can be rescheduled. To reduce the peak load, you must schedule the 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, this mechanism is a temporary solution. See 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 operating system commands and guidelines on performance tuning of the network or operating system, refer to the documentation provided by the operating system vendor.

## 2.4 Tuning Java Virtual Machines (JVMs)

How you tune your Java virtual machine (JVM) greatly affects the performance of Oracle Fusion Middleware and your applications.

For more information on tuning your JVM, see Tuning Java Virtual Machines (JVM) in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*.

## 2.5 Tuning the WebLogic Server

Tune the WebLogic Server to match your application needs.

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

## 2.6 Tuning Database Parameters

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

> **Note:**
>
> The information in these topics is a subset of database tuning information for Fusion Middleware. Make sure that you have also reviewed the *Database Performance Tuning Guide*.
>
> Always review the tuning guidelines in your database-specific vendor documentation.

- Tuning Database Parameters
- Tuning Database Files
- Tuning Automatic Segment-Space Management (ASSM)

## 2.6.1 Tuning Database Parameters

The following tables provide common `init.ora` parameters and their descriptions. Follow these guidelines to set the database parameters. Ultimately, however, the database administrator must monitor the database health and tune parameters based on the need.

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

The `sga_target`, `pga_aggregate_target`, and `processes` parameters from Table 2-2 are examples of such parameters that must be tuned based on the System Global Area (SGA) and Parent Global Area (PGA) advisories and looking into the number of open processes during peak load.

**Table 2-2 Important Oracle 12c Database Tuning Parameters**

| Parameter | Description | Tuning Recommendation |
|---|---|---|
| `audit_trail`<br>**Default**: `DB` | Enables or disables database auditing. | Set to `NONE` if there is NO policy to audit database activity. Enabling auditing can impact performance. |
| `plsql_code_type`<br>**Default**: `INTERPRETED` | Compilation mode for PL/SQL library units. Possible modes are as follows:<br>• `INTERPRETED`: PL/SQL library units are compiled to PL/SQL byte code format and executed by the PL/SQL interpreter engine.<br>• `NATIVE`: PL/SQL library units are compiled to native (machine) code. Such modules are executed natively without incurring any interpreter impacts. | Set to `NATIVE`. |

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

| Parameter | Description | Tuning Recommendation |
|---|---|---|
| `nls_sort`<br>**Default**: Derived from `NLS_LANGUAGE` | Collating sequence for `ORDER BY` queries.<br>• If the value is a named linguistic sort, the collating sequence is based on the order of the defined linguistic sort. Most languages supported by the `NLS_LANGUAGE` parameter also support a linguistic sort with the same name.<br>• If the value is set to `BINARY`, then the collating sequence is based on the numeric value of characters. This requires fewer system resources. | Set to `BINARY`. |
| `open_cursors`<br>**Default**: `50` | Maximum number of open cursors that a session can have at once. Open cursors are handles to private SQL areas.<br>The value of `OPEN_CURSORS` must be high enough to prevent your application from running out of open cursors. | Increase to `500`. |
| `session_cached_cursors`<br>**Default**: `50` | Number of session cursors to cache. Repeated parse calls of the same SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Subsequent parse calls locate the cursor in the cache. However, they do not reopen the cursor. Oracle uses a **least recently used** algorithm to remove entries in the session cursor cache to make room for new entries when needed.<br>This parameter also constrains the size of the PL/SQL cursor cache, which PL/SQL uses to avoid having to reparse as statements are reexecuted. | Increase to `500`. |
| `_b_tree_bitmap_plans`<br>**Default**: `TRUE` | Enables or disables the use of bitmap access paths for b-tree indexes. | Set to `FALSE`. |
| `processes`<br>**Default**: `100` | Maximum number of operating system processes that can be connected to the Oracle database concurrently. The value of this parameter must account for Oracle the background processes.<br>The `SESSIONS` parameter is deduced from this value. | For most systems, increasing to `1500` must suffice.<br>For a large-scale system, such as databases with a large number of users, the recommended value is `5000`. |

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

| Parameter | Description | Tuning Recommendation |
|---|---|---|
| Memory_target | Oracle system-wide usable memory. The database tunes memory to the MEMORY_TARGET value, reducing or enlarging the SGA and PGA as needed. | Consider setting to NONE. Then set the SGA and PGA targets separately as setting MEMORY_TARGET does not allocate sufficient memory to SGA and PGA as needed. |
| sga_target<br>**Default**: 0 | A non-zero value enables Automatic Shared Memory Management. This can simplify configuration and improve performance. | For small systems, use a minimum of 2 GB.<br>For large systems, set it to 18 GB. |
| pga_aggregate_target<br>**Default**: 0 | Target aggregate PGA memory available to all server processes attached to the instance. | For small systems, use a minimum of 1 GB.<br>For large systems, set it to 8 GB. |
| Disk_asynch_io<br>**Default**: TRUE | Controls whether I/O to data files, control files, and log files is asynchronous. It decides what parallel server processes can overlap I/O requests with CPU processing during table scans. | Set to FALSE only if your platform does not support asynchronous I/O. |
| Filesystemio_options<br>**Default**: None | I/O operations for file system files. | Set to SETALL. |
| Secure_Files<br>**Default**: PERMITTED | How to store LOB objects from tables. | Set to ALWAYS. |
| parallel_max_servers<br>**Default**: PARALLEL_THREADS_PER_CPU*CPU_COUNT*concurrent_parallel_users*5 | Maximum number of parallel execution processes and parallel recovery processes for an instance.<br>As the demand increases, the Oracle database increases the number of processes from the number created at instance startup to this value. | Set to 12. |
| job_queue_processes<br>**Default**: 1000 | Maximum number of job slaves per instance that can be created for the execution of DBMS_JOB jobs and Oracle Scheduler (DBMS_SCHEDULER) jobs. | Set to 12. |
| shared_servers<br>**Default**: 0 (or) 1 | Number of server processes that you want to create when an instance is started. | Set to 0. |

The table below describes the important inti.ora Database Tuning Parameters.

**Table 2-3 Important inti.ora Oracle 12c Database Tuning Parameters**

| Database Parameter | Description |
|---|---|
| AUDIT_TRAIL | If there is no policy to audit database activity, consider setting this parameter to NONE. Enabling auditing can impact performance. |

**Table 2-3 (Cont.) Important inti.ora Oracle 12c Database Tuning Parameters**

| Database Parameter | Description |
|---|---|
| MEMORY_MAX_TARGET | Maximum value to which a database administrator can set the MEMORY_TARGET initialization parameter. |
| MEMORY_TARGET | Consider setting to NONE. Set SGA and PGA separately as setting the 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 monitor the production database daily and adjust SGA and PGA accordingly.<br>If the database server has more memory, consider setting the PGA_AGGREGATE_TARGET to a value higher than 1G, based on usage needs. |
| SGA_MAX_SIZE | Consider setting the MEMORY_TARGET instead of setting SGA and the PGA separately. |
| SGA_TARGET | Consider using a value of 2G initially and then monitor the production database daily and adjust SGA and PGA accordingly.<br>If the database server has more memory, consider setting the SGA_TARGET to a value higher than 2G, based on usage needs. |

In addition, set a minimum value for SHARED_POOL_SIZE and DB_CACHE_SIZE to minimize frequent resizing.

## 2.6.2 Tuning Database Files

In addition to tuning the database parameters, the database administrator must configure the REDO logs, UNDO table space, and TEMP table spaces to meet the demands of the database workload. The recommendations here are intended to provide initial guidance in these areas.

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

- Configuring REDO Logs
- Configuring UNDO Tablespace
- Configuring TEMP Tablespace

### 2.6.2.1 Configuring REDO Logs

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

Depending on your storage configuration and performance characteristics, redistribute the REDO logs to optimize I/O performance. The REDO log files must be placed on a disk separately from the data files to improve the I/O performance.

See Managing the REDO Log*Oracle Database Administrator's Guide*

### 2.6.2.2 Configuring UNDO Tablespace

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

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

### 2.6.2.3 Configuring TEMP Tablespace

Oracle recommends the use of locally managed temporary tablespaces with the allocation type set to `UNIFORM` extents and the default size of 1 MB.

For tuning `TEMP` tablespaces for SOA, see Tuning Temporary Tablespaces for SOA.

## 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`.

Oracle recommends to specify the allocation type to `SYSTEM`.

See Free Space Management and Specifying Segment Space Management in Locally Managed Tablespaces in *Oracle Database Administrator's Guide*.

# 2.7 Reusing Database Connections

It is important to tune the connection pool attributes in the JDBC data sources in your WebLogic Server domain correctly to improve application and system performance.

Creating a database connection is a resource-intensive process in any environment. Typically, a connection pool starts with a few connections. As client demands for more connections grow, there will not be enough in the pool to fulfill the requests. WebLogic Server creates more 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. This is 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 preproduction performance testing.

When the load is uneven, and has high number of connections at peak load than at typical load, set the initial number of connections equal to your typical load. In addition, set 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.

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

## 2.8 Enabling Data Source Statement Caching

Statement caching improves performance by caching executable statements that are used repeatedly.

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

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

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

Each connection in a data source has its own individual cache of prepared and callable statements used on the connection. However, you configure statement cache options as per the data source. That is, the statement cache for each connection in a data source uses the statement cache options specified for the data source. Each connection caches its 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. Analyze your database 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.

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

# 2.9 Controlling Concurrency

Limiting concurrency at multiple layers of the system to match specific usage needs can greatly improve performance.

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

- Setting Server Connection Limits
- Configuring Connection Pools
- Tuning the WebLogic Server Thread Pool

## 2.9.1 Setting Server Connection Limits

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

For details on modifying the `httpd.conf file`, see Configuring Oracle HTTP Server in *Oracle Fusion Middleware Administering Oracle HTTP Server*.

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. There are several Oracle HTTP Server tuning parameters related to connection limits that must be tuned based on the expected client load. See Tuning Oracle HTTP Server for details on setting server connection limits and a complete list of tunable parameters.

- Setting MaxClients / ThreadsPerChild
- Setting KeepAlive
- Tuning HTTP Server Modules

### 2.9.1.1 Setting MaxClients / ThreadsPerChild

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

The `MaxClients` parameter 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 the system is not resource-saturated and the user

population is more than 150 concurrent HTTP connections, 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 requires sufficient concurrent `httpd` server processes to handle both active and idle connections. When you specify `MaxClients` to act as a throttle for system concurrency, 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 few 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 these properties must be set appropriately to achieve the number of connections as specified by `MaxClients`. See Table 6-1 for a description of all the HTTP configuration properties.

### 2.9.1.2 Setting KeepAlive

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

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

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

### 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 the `mod_wl_ohs` module are available in the `mod_wl_ohs.conf` file in the `config` directory.

See Understanding Oracle HTTP Server Modules in *Oracle Fusion Middleware Administering Oracle HTTP Server*.

## 2.9.2 Configuring Connection Pools

Connection pooling is configured and maintained per Java runtime. Connections are not shared across different runtimes. To use connection pooling, no configuration is required. Configuration is necessary only if pooling needs to be customized. For example; control the size of the pools and types of connections to be pooled.

You configure connection pooling by using several system properties at program startup time. These are system properties, not environment properties and they affect all connection pooling requests.

For applications that use a database, performance can improve when the connection pool that is associated with a data source limits the number of connections. 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. Thus, 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 the `MaxCapacity` attribute is set to 15. For best performance, specify a value for the `MaxCapacity` attribute 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. Configure the database to allow at least 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.

For connection pool options, see *JDBC Data Source: Configuration: Connection Pool* in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help* and Tuning Data Source Connection Pool Options in *Oracle Fusion Middleware Administering JDBC Data Sources for Oracle WebLogic Server*.

## 2.9.3 Tuning the WebLogic Server Thread Pool

By default, WebLogic Server uses a single thread pool. All types of work are executed in this thread pool. WebLogic Server uses work managers to prioritize work based on rules that you can define, and runtime 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.

The WebLogic Server thread pool is sized automatically and hence in most situations you do not need to tune it. 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 executed 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 is given a higher priority over another.
- A response time goal is required.
- A minimum thread constraint is 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 are 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.

> **Note:**
>
> For details 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 Tuning Performance of Oracle WebLogic Server*
> - Thread Management in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*
> - MDB Thread Management in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*
> - Using Work Managers to Optimize Scheduled Work in *Oracle Fusion Middleware Administering Server Environments for Oracle WebLogic Server*
> - Avoiding and Managing Overload in *Oracle Fusion Middleware Administering Server Environments for Oracle WebLogic Server*

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 the scope of the application and the

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.

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.

## 2.10 Setting Logging Levels

The amount of information that is logged can have a significant impact on the performance.

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.

After you set the application and server logging levels, ensure that the debugging properties or other application level debugging flags are set correctly 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 has specific recommendations for logging levels.

# 3
# Performance Planning

A clearly defined plan for achieving your performance objectives is essential for deciding what to trade for higher performance.

- About Performance Planning
  To maximize performance, you must monitor, analyze, and tune all the components that are used by your applications.
- 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.

## 3.1 About Performance Planning

To maximize performance, you must monitor, analyze, and tune all the components that are used by your applications.

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

## 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.

- Step 1: Defining Your Performance Objectives
- Step 2: Designing Applications for Performance and Scalability
- Step 3: Monitoring and Measuring Your Performance Metrics

### 3.2.1 Step 1: Defining 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.

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 to determine 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 must focus on maximum performance requirements.
- The configuration of high availability architecture to address peak usage and response times. For more information on implementing high availability features in Oracle Fusion Middleware applications, see Introduction and Roadmap in *Oracle Fusion Middleware High Availability Guide*.
- The ability to interoperate between domains, use legacy systems, support legacy data.
- Development, implementation, and maintenance costs.

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

- Defining Operational Requirements
- Identifying Performance Goals
- Understanding User Expectations
- Conducting Performance Evaluations

### 3.2.1.1 Defining 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 Identifying 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 application components are expected to meet, such as:

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

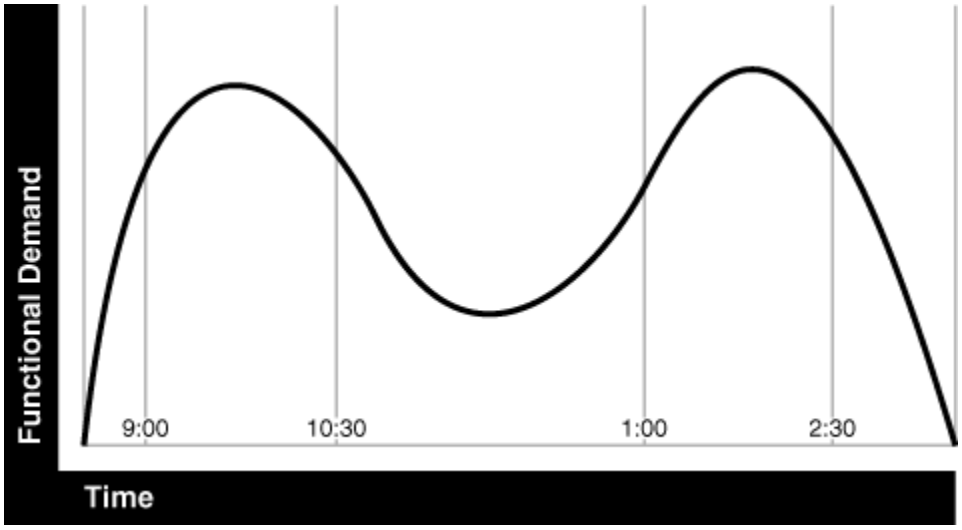
### 3.2.1.3 Understanding User Expectations

Application developers, database administrators, and system administrators must be careful to set appropriate performance expectations for users. When the system carries out a complicated operation, response time is slower than while performing a simple operation.

For example, ensure that 90% of the response time is not greater than 5 seconds and the maximum response time for all is 20 seconds. Usually, it's not that simple. Application may include various operations with differing characteristics and acceptable response time. Set measurable goals for each of these operations.

Determine how variances in the load can affect the response time. For example, users might access the system heavily between 9:00 am and 10:00 am and then again between 1:00 pm and 2:00 pm, as illustrated by the graph in Figure 3-1. If the peak load occurs regularly, for example, daily or weekly, it is advised to configure and tune systems to meet the peak load requirements. Accessing application in off-time gives better response time than accessing it during peak-time. If your peak load is infrequent, higher response times at peak loads must be expected for the cost savings of smaller hardware configurations.

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




### 3.2.1.4 Conducting 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 that 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 the actual resource consumption for a range of loads, you can conduct objective scalability studies and from these reports predict the

resource requirements for anticipated load volumes. For details on Monitoring, see, Monitoring .

## 3.2.2 Step 2: Designing Applications for Performance and Scalability

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

## 3.2.3 Step 3: Monitoring and Measuring 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 the server activity, watch trends, diagnose system bottlenecks, debug applications with performance problems and gather data that can assist you in tuning the system.

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

# 4
# Monitoring

Oracle Fusion Middleware provides a variety of technologies and tools that monitor server and application performance.

- About Oracle Fusion Middleware Management Tools
  Monitoring enables you to evaluate server activity, watch trends, diagnose system bottlenecks, debug applications with performance problems, and gather data that can assist in tuning the system.
- Oracle Enterprise Manager Fusion Middleware Control
  Fusion Middleware Control is a web browser-based, graphical user interface that you can use to monitor and administer your domain.
- 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.
- 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.
- 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.
- DMS Spy Servlet
  The DMS Spy Servlet provides access to DMS metric data from a web browser.
- Native Operating System Performance Commands
  Each operating system has native tools and utilities that can be useful for monitoring purposes.
- 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.

## 4.1 About Oracle Fusion Middleware Management Tools

Monitoring enables you to evaluate server activity, watch trends, diagnose system bottlenecks, debug applications with performance problems, and gather data that can assist in tuning the system.

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

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

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



### 4.1.1 Measuring Your Performance Metrics

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

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

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

If you are new to Oracle Fusion Middleware or if you need additional information about monitoring your environment by using the Performance Summary pages, see Viewing the Performance of Oracle Fusion Middleware in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*. In addition, the Fusion Middleware Control online help provides definitions and other information about specific performance metrics that are available on its management and monitoring pages.

## 4.2 Oracle Enterprise Manager Fusion Middleware Control

Fusion Middleware Control is a web browser-based, graphical user interface that you can use to monitor and administer your domain.

It can manage an Oracle WebLogic Server domain with its Administration Server, one or more Managed Servers, clusters, the Oracle Fusion Middleware components that are installed, configured, and running in the domain, and the applications that you deploy.

See Getting Started Using Oracle Enterprise Manager Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

## 4.3 Oracle WebLogic Server Administration Console

Oracle WebLogic Server Administration Console is a web browser-based, graphical user interface that you use to manage an Oracle WebLogic Server domain.

It is accessible from any supported web browser with network access to the Administration Server.

See Getting Started Using Oracle WebLogic Server Administration Console in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

**Additional WebLogic Server Console Resources:**

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

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

## 4.4 WebLogic Diagnostics Framework (WLDF)

The WebLogic Diagnostic Framework (WLDF) is a monitoring and diagnostic framework that can collect diagnostic data that servers and applications generate.

The WLDF can be configured to collect the data and store it in various sources, including log records, data events, and harvested metrics.

See Understanding the Diagnostic Framework in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

> **Note:**
>
> For details 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.

See Getting Started Using the Oracle WebLogic Scripting Tool (WLST) in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

# 4.6 DMS Spy Servlet

The DMS Spy Servlet provides access to DMS metric data from a web browser.

Data that is created and updated by DMS-enabled applications and components is accessible through the DMS Spy Servlet.



## 4.6.1 Viewing Performance Metrics Using the Spy Servlet

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

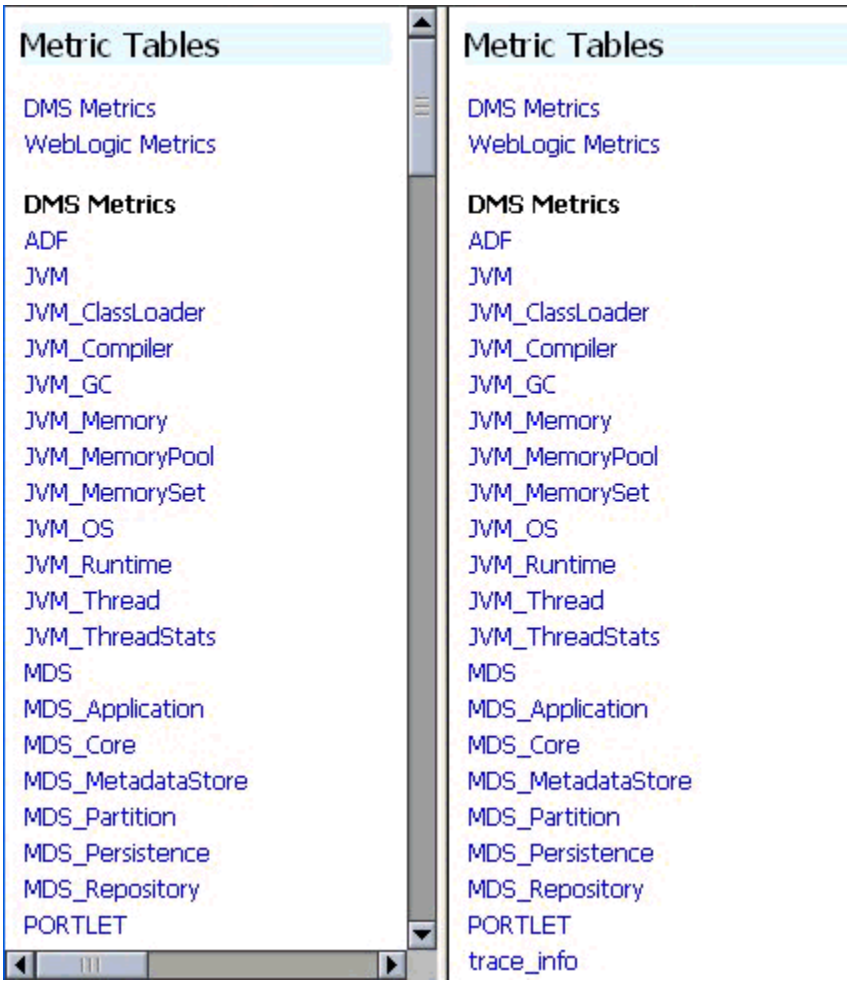
The DMS web application is deployed by default as part of a JRF-enabled server instance. The URL is: `http://host:port/dms/Spy`.

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

## 4.6.2 Using the DMS Spy Servlet

Figure 4-1 shows the initial page of the Spy servlet: both sides show the same list of metric tables.

**Figure 4-1 Spy Servlet Page - Metrics Tables**

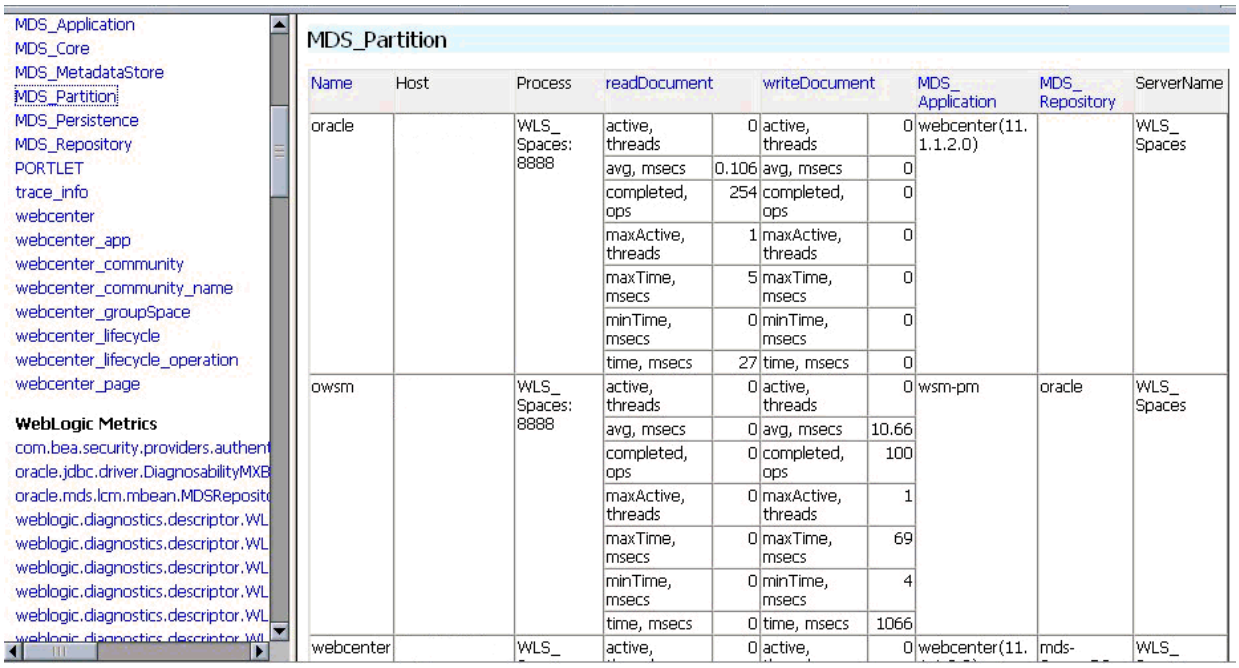



The Spy servlet can display metric tables for WebLogic Server and also for non-Java EE components that are deployed.

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

To view the contents of a metric table, click the table name. For example, Figure 4-2 shows the `MDS_Partition` table.

**Figure 4-2 MDS Partition Table**




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

## 4.7 Native Operating System Performance Commands

Each operating system has native tools and utilities that can be useful for monitoring purposes.

Native operating system commands enable you to gather and monitor system activity information. For example CPU utilization, paging activity, swapping, and so on.

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

## 4.8 Network Performance Monitoring Tools

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

For details on network performance monitoring, refer to your operating system documentation.

5

# Using the Oracle Dynamic Monitoring Service

The Oracle Dynamic Monitoring Service (DMS) publishes component performance data.

- 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.
- About DMS Availability
  DMS functionality is available on all certified Java EE servers.
- About DMS Architecture
  It is important to understand the components of DMS and how they interact with other Oracle Fusion Middleware components.
- Viewing DMS Metrics
  Oracle Fusion Middleware components are instrumented with DMS metrics to collect information that developers, system administrators, and support analysts can use to analyze system performance or monitor system status.
- Accessing DMS Metrics with WLDF
  WLDF can be configured to collect MBean data when specific conditions are met, for monitoring purposes.
- About DMS Execution Context
  The DMS execution context is the mechanism by which requests (such as RMI requests) can be uniquely identified and thus tracked as they flow through the system.
- DMS Tracing and Events
  The DMS tracing feature can be used to diagnose issues or collect specific data at a specific time for a specific set of criteria.
- DMS Best Practices
  Implement the following best practices when you use DMS metrics.

## 5.1 About Dynamic Monitoring Service (DMS)

The Oracle Dynamic Monitoring Service (DMS) enables Oracle Fusion Middleware components to provide administration tools, such as Oracle Enterprise Manager, with data regarding the component's performance, state, and on-going behavior.

Fusion Middleware components push data to DMS and in turn DMS publishes that data through a range of different components. DMS measures and reports metrics, traces events and system performance, and provides a context correlation service for these components.

- Understanding Common DMS Terms and Concepts

## 5.1.1 Understanding Common DMS Terms and Concepts

There are common DMS terms and concepts related to DMS Senors, DMS Nouns, and DMS Tracing and Events.

- DMS Sensors
- DMS Nouns
- DMS Tracing and Events

### 5.1.1.1 DMS Sensors

DMS *sensors* measure performance data and enable DMS to define and collect a set of metrics. Certain metrics are always included with a sensor and others are optional.

- DMS PhaseEvent Sensors
- DMS Event Sensors
- DMS State Sensors
- Sensor Naming Conventions

#### 5.1.1.1.1 DMS PhaseEvent Sensors

A DMS *PhaseEvent sensor* measures the time spent in a specific section of code that has a beginning and an end. Use a PhaseEvent sensor to track time in a method or in a block of code.

DMS can calculate optional metrics that are associated with a PhaseEvent, including the average, maximum, and minimum time that is spent in the PhaseEvent sensor.

Table 5-1 lists the metrics that are available with PhaseEvent sensors.

**Table 5-1 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. |

**Table 5-1 (Cont.) DMS PhaseEvent Sensor Metrics**

| Metric | Description |
|---|---|
| *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 changes 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. |

### 5.1.1.1.2 DMS Event Sensors

A DMS *event sensor* counts system events. Track system events through a DMS event sensor that has a short duration, or where the occurrence of the event is of interest.

Table 5-2 describes the metric that is associated with an event sensor.

**Table 5-2 DMS Event Sensor Metrics**

| Metric | Description |
|---|---|
| *sensor_name*.count | Specifies the number of times the event has occurred since the process started. *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. |

### 5.1.1.1.3 DMS State Sensors

A DMS *state sensor* tracks the value of Java primitives or the content of a Java object. Supported types include integer, double, long, and object. Use a state sensor when you want to track the system status information or when you need a metric that is not associated with an event. For example, use state sensors to track queue lengths, pool sizes, buffer sizes, or host names. You assign a precomputed value to a state sensor.

Table 5-3 describes the state sensor metrics. State sensors support a default metric `value`, as well as optional metrics. The optional `minValue` and `maxValue` metrics only apply for state sensors if the state sensor represents a numeric Java primitive (of type integer, double, or long).

**Table 5-3 DMS State Sensor Metrics**

| Metric | Description |
|---|---|
| *sensor_name*.value | Specifies the metric value for *sensor_name*, by 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 for this *sensor_name* since startup.<br>Optional metric. |

### 5.1.1.1.4 Sensor Naming Conventions

The following list describes the DMS sensor naming conventions:

- Sensor names must be descriptive, but not redundant. Sensor names should not contain any part of the noun name hierarchy, or type, as it is redundant.
- Sensor names must avoid containing the value for the individual metrics.
- Where multiple words are required to describe a sensor, the first word must start with a lowercase letter, and the following words must start with uppercase letters. For 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, use an alternative delimiter, such as **,** or _, which does not appear anywhere in the path; it 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 by using the string:

  ```
  ,default,WEBs,defaultWebApp,JSPs,example/jsp/num/numguess.jsp,service
  ```

  where the delimiter is the **,**character.
- The Event sensor and PhaseEvent sensor names should have the form *verbnoun*. For example, `activateInstance` and `runMethod`. When a PhaseEvent monitors a function, method, or code block, it must be named to reflect the task performed as clearly as possible.
- The name of a state sensor must be a noun, possibly preceded by an adjective, which describes the semantics of the value that is tracked with this state sensor. For example, `lastComputed`, `totalMemory`, `port`, `availableThreads`, `activeInstances`.

- To avoid confusion, do not name sensors with strings such as `.time`, `.value`, or `.avg`, which are names of sensor metrics, as shown in Table 5-1, Table 5-2, and Table 5-3.

### 5.1.1.2 DMS Nouns

DMS **nouns** organize performance data. Sensors, with their associated metrics, are organized in hierarchy according to nouns. Nouns enable you to organize DMS metrics in a manner comparable to a directory structure in a file system. For example, nouns can represent classes, methods, objects, queues, connections, applications, databases, or other objects that you want to measure.

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

- General DMS Naming
- General DMS Naming Conventions and Character Sets
- Noun and Noun Type Naming Conventions

#### 5.1.1.2.1 General DMS Naming

A **noun name** is a string, which does not include 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. For example, `/dmsDemo/BasicBinomial/"{http://mynamespace/}JAXWSHelloService"` is a noun full name.

A **sensor name** is a string, which does not include 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. For example, `/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 must be as compact as possible. When you define noun and sensor names, avoid special characters such as white space, slashes, periods, parenthesis, commas, and control characters.

Table 5-4 shows the DMS replacement for special characters in names.

**Table 5-4 Replacement for Special Characters in DMS Names**

| Character | DMS Replacement Character |
|---|---|
| Space character | Underscore character: _ |
| 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 must be unique.
- A noun name must identify a specific entity of interest.
- Noun types should have names that clearly reflect the set of metrics that are 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 must start with a capital letter to distinguish them from other DMS names. All nouns of a given type must contain the same set of sensors.
- The noun naming scheme uses a ***/*** as the root of the hierarchy, with each noun acting as a container under the root or under its parent noun.

### 5.1.1.3 DMS Tracing and Events

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

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

**Table 5-5 DMS Tracing and Event Terminology**

| DMS Term | Definition |
|---|---|
| Condition | A **condition** is the logic behind a condition filter. It determines which events might pass through a filter, based on the rules defined in the condition. Every condition filter has zero or one root condition, but conditions might 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 that is associated with a sensor or noun event.<br>• Context Condition: operates on the values currently set within the current Execution Context.<br>See DMS Tracing and Events. |
| Destination | A **destination** implements a mechanism for reacting to events that are passed to it. For example, a destination logs events to a file, sends transformed copies of events to the Java Flight Recorder, renders information gathered from incoming events as data in an MBean. |
| Event Route | An **event route** connects a filter to a destination. Event routes can be enabled or disabled. |
| Filter | An event tracing **filter** selectively passes a subset of all possible DMS runtime events. Filters can be configured with rules that determine the events that are passed and the events that 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>See DMS Tracing and Events. |
| Listener | A DMS **listener** is also known as the destination. See Configuring Destinations. |

# 5.2 About 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 operates in JSE applications and standalone C applications.

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

# 5.3 About DMS Architecture

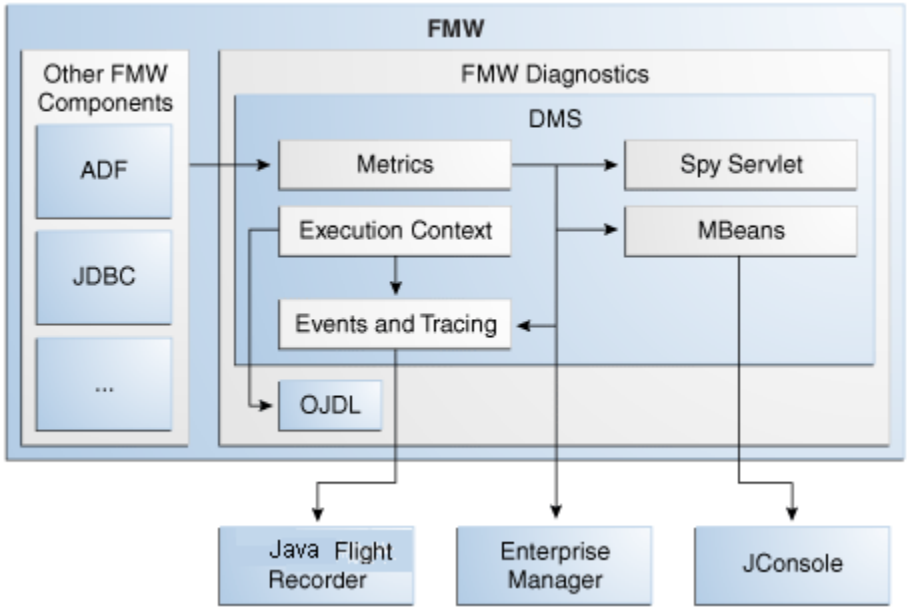
It is important to understand the components of DMS and how they interact with other Oracle Fusion Middleware components.

DMS consists of the following features:

- **DMS Metrics**: The DMS metrics feature provides Java and C APIs, which the Oracle Fusion Middleware components use for instrumenting code with performance measurements and other useful state metrics.
- **Execution Context**: Execution Context supports the maintenance and propagation of a specific context structure throughout the Oracle stack. By exploiting the propagated context structure, Oracle Fusion Middleware components can record diagnostic information (such as log records) that can be correlated between different components and products running on the same or different servers and hosts. See About DMS Execution Context.
- **Events and Tracing**: Event Tracing enables you to configure live tracing with no restarts. DMS metrics that are updated by using Oracle Fusion Middleware products must be traced by using the DMS Event Tracing feature. The system has been designed to facilitate not only tracing, but also to support the other functionality that is driven from DMS activity.

Figure 5-1 shows the components of DMS and how they interact with other Oracle Fusion Middleware components. The 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 to collect information that developers, system administrators, and support analysts can use to analyze system performance or monitor system status.

The Fusion Middleware Control online help provides information on each of the specific metrics. See Viewing the Performance of Oracle Fusion Middleware in *Oracle Fusion Middleware Administering Oracle Fusion Middleware* for information on accessing metric information.

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

You can use various tools to view the DMS metrics.

- Viewing Metrics By Using the Spy Servlet
- Viewing Metrics with WLDF (WebLogic Diagnostic Framework)
- Viewing Metrics with WLST (Oracle WebLogic Server)
- Viewing Metrics with JConsole
- Viewing Metrics with Oracle Enterprise Manager

## 5.4.1 Viewing Metrics By Using the Spy Servlet

The Spy Servlet is part of the DMS Application that is deployed by default on JRF-extended installations. The Spy Servlet is launched from `http://<host>:<port>/dms/Spy`. The default port for WebLogic is `1521`.

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

See DMS Spy Servlet.

> **Note:**
>
> The Spy Servlet is secured by using standard Java EE declarative security in the web-application's `web.xml` file, and access is granted only 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. 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 and they are detailed in the table below.

**Table 5-6 DMS Commands**

| Use this command... | To do this... |
|---|---|
| `displayMetricTableNames()` | List the names of the available metric tables.<br>If there are many metric tables, consider using the `outputfile` parameter with `displayMetricTableNames()`. It is useful when the output is expected to be large. When `displayMetricTableNames()` has the `outputfile` parameter, it returns null to the script instead of the whole output. This prevents the command from running out of memory.<br>**NOTE**: The command syntax for `displayMetricTableNames()` differs slightly for system components (such as OHS). After you connect WLST to Node Manager by using the `nmConnect()` command, you must specify both the server name and the server type explicitly.<br>For example:<br>`displayMetricTableNames(servertype="OHS", servers="ohs1")` |
| `displayMetricTables()` | Show the content of the DMS metric tables.<br>If you have many DMS metric tables, consider using the `outputfile` parameter with `displayMetricTables()`. This is useful when the output is expected to be large. When `displayMetricTables()` has the `outputfile` parameter, it returns null to the script instead of the whole output. This prevents the command from running out of memory.<br>**NOTE**: The command syntax for `displayMetricTables()` differs slightly for system components (such as OHS). After you connect WLST to Node Manager by using the `nmConnect()` command, you must specify both the server name and the server type explicitly.<br>For example:<br>`displayMetricTables(servertype="OHS", servers="ohs1")` |
| `dumpMetrics()` | Display metrics in the internal format. Valid formats for the dumpMetrics command include raw, xml, and pdml.<br>If you have many DMS metric tables, consider using the `outputfile` parameter with `dumpMetrics()`. This is useful when the output is expected to be large. When `dumpMetrics()` has the `outputfile` parameter, it returns null to the script instead of the whole output. This prevents the command from running out of memory.<br>**NOTE**: The command syntax for `dumpMetrics()` differs slightly for system components (such as OHS). After you connect WLST to Node Manager by using the `nmConnect()` command, you must specify both the server name and the server type explicitly.<br>For example:<br>`dumpMetrics()(servertype="OHS", servers="ohs1")` |

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

For details on using these commands, see the following:

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

### 5.4.4 Viewing Metrics with JConsole

To provide a standards-based way to access metrics, DMS exposes them through MBeans. An MBean is created and registered for each type 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 Accessing DMS Metrics with WLDF. 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 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 the on-going behavior. The metrics are automatically enabled; there is no need to set options or perform any extra configuration to collect them. See Oracle Enterprise Manager Fusion Middleware Control.

## 5.5 Accessing DMS Metrics with WLDF

WLDF can be configured to collect MBean data when specific conditions are met, for monitoring purposes.

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 email and JMX notifications when a condition is triggered.

The following steps describe how to configure WLDF to send an email notification by using the WebLogic Administration Console:

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

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

6. Select **Use a manual reset alarm** and click **Next**.

   The manual reset option means that once an email is triggered, you must reset the watch by using the WebLogic Administration Console.

7. Select the email 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 by using the WebLogic Administration Console.

For details on using WLDF to harvest and monitor MBean data, see Configuring the Harvester for Metric Collection in *Oracle Fusion Middleware Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

# 5.6 About DMS Execution Context

The DMS execution context is the mechanism by which requests (such as RMI requests) can be uniquely identified and thus tracked as they flow through the system.

It also provides the means by which context information can be communicated between cooperating Fusion Middleware components involved in fulfilling requests.

- DMS Execution Requests and Subtasks
- DMS Execution Context Usage
- DMS Execution Context Communication

## 5.6.1 DMS Execution Requests and Subtasks

The DMS execution context has been developed with the understanding that a single request (or task) might create many subtasks that are coordinated to complete the request or root task. Consider the following examples of requests and their associated subtasks:

1. A request sent directly to Oracle WebLogic Server from a browser:
   - Root task only on Oracle WebLogic Server
2. A request sent through Oracle Server (acting as a reverse proxy) to Oracle WebLogic Server:
   - Root task on Oracle Server
   - Single sub-task on Oracle WebLogic Server

3. A request sent from an Oracle Server (acting as a reverse proxy) to an Oracle WebLogic Server that requires invocation of two remote web services from an Oracle WebLogic Server to fulfill the request:
    - Root task on an Oracle Server
    - Single sub-task on an Oracle WebLogic Server
    - Two sub-subtasks, one on each web service

A DMS execution context is composed of the following:

- A unique identifier, the Execution Context ID (ECID).

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

  The 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 a request is sent from an Oracle Server (acting as a reverse proxy) to an Oracle WebLogic Server and the server requires invocation of two remote web services from Oracle WebLogic Server.

1. Root task on Oracle Server:
    - New ECID = B5C094FA...BE4AE8
    - Root RID = 0
2. Single subtask on Oracle WebLogic Server:
    - Same ECID = B5C094FA...BE4AE8
    - Sub-task RID = 0:1
3. Two subtasks, 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 that are associated with that task by querying the log files for messages that contain that ECID.

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

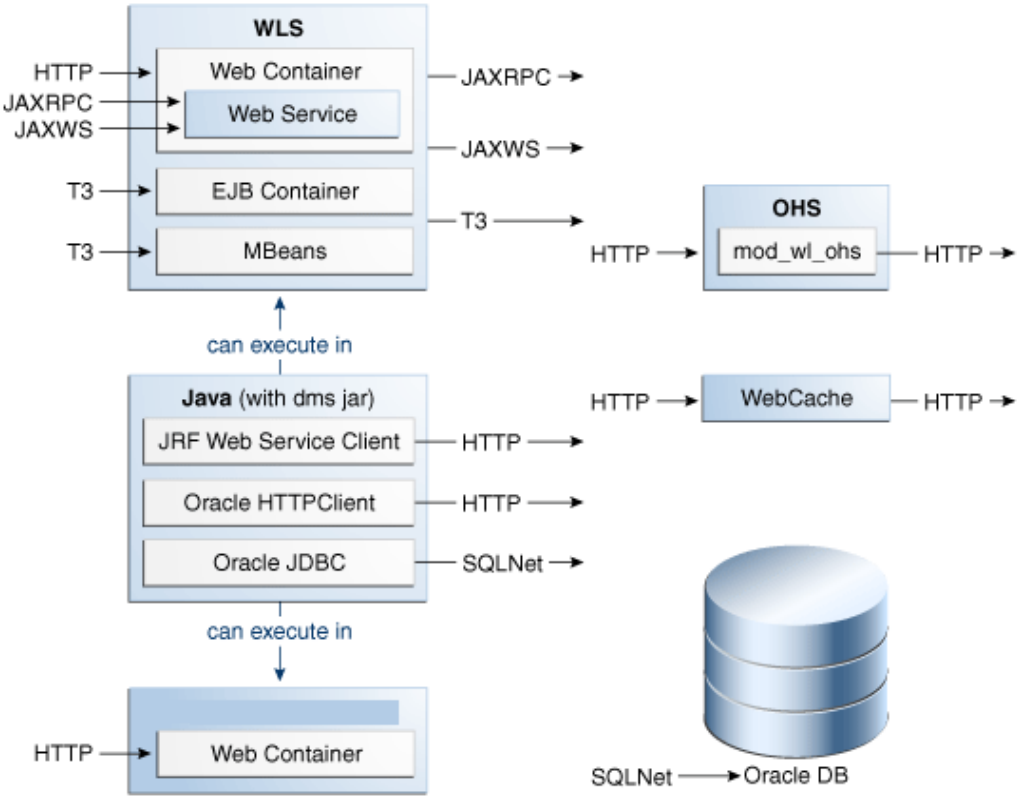
```
displayLogs(ecid="B5C094FA...BE4AE8");
```

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

## 5.6.3 DMS Execution Context Communication

Figure 5-2 shows the components that cooperate 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

The DMS tracing feature can be used to diagnose issues or collect specific data at a specific time for a specific set of criteria.

DMS can selectively trace the following:

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

- Context events (start, stop)
- Events (start, stop)

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

1. Filter Configuration

   Defines the rules that select the events that are of interest

2. Destination Configuration

   Defines how the events are used

3. eventRoute Configuration

   Defines which filters are wired to which destinations

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

The configuration can be modified by 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.

See Configuring Selective Tracing Using WLST in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

The following types of filter rules are supported:

- Event Type Conditions

  Used to identify if an event was triggered from the `START` or `STOP` of a `PHASE_SENSOR`

- Context Type Conditions

  Used to identify if the event was generated from a unit of work whose context contains a value (for example, `USER`)

- Noun Type Conditions

  Used to identify if the event was triggered from a sensor whose noun is of a specific type (for example, `JDBC_CONNECTION`)

- Logical `AND` and `OR` combinations of the conditions mentioned

- Configuring the DMS Event System
- Configuring Destinations
- Understanding the Format of DMS Events in Log Messages
- Understanding DMS Event Actions

## 5.7.1 Configuring the DMS Event System

Configuration is recorded in each server `dms_config.xml` file. MBean updates can be made at runtime by using the 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 the DMS event configuration on your system , use the following command:

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

The resulting output looks similar to:

```
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
```

- Adding and Editing Filters
- Adding and Editing Destinations
- Adding and Editing Event Routes
- Compound Operations

## 5.7.1.1 Adding and Editing Filters

Filters define the rules that select the events that 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 that are associated with JDBC operations are performed on nouns of types whose names begin with `JDBC_`.

If the rule must be modified, the filter must 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 by using either the shortened or longer name.

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 works, but is discouraged.

**Table 5-7 DMS Operators**

| Noun Type Operators | Details |
|---|---|
| equals, eq | notEquals, ne |
| contains | in |
| startsWith, sw | - |

| Context Operators | Details |
|---|---|
| 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 details on 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 logs the event, a different destination might transform an event and pass it to another system for further processing.

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

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

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

The types of destination available, and their configuration options, are described in Configuring Destinations. 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 create a destination.

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

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 that are 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 by using a single command (rather than using two separate commands as shown in Adding and Editing Event Routes).

> **Note:**
>
> 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 by 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 several types of destinations.

- LoggerDestination
- MBean Creator Destination
- Request Tracker Destination
- Java Flight Recorder Destination

### 5.7.2.1 LoggerDestination

**Table 5-8 Logger Destination**

| Properties | Details |
|---|---|
| **Description** | The *LoggerDestination* writes each event to the associated logger. |
| **Implementing Class** | *oracle.dms.trace2.runtime.LoggerDestination* |
| **Properties** | |

**Table 5-8 (Cont.) Logger Destination**

| Properties | Details |
|---|---|
| `loggerName` | The name of the ODL logger to which events are 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 Static Loggers and Handlers). 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 Dynamic Loggers and Handlers).

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

- Static Loggers and Handlers
- Dynamic Loggers and Handlers
- Default Locations of the logging.xml File
- Using a CLI Command to Query the Trace Log File

### 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`. It allows you to define the name of the log file, the maximum size, format, file rotation, and policies.

Oracle recommends using commands (like the example here) 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 details on logging configuration, see Managing Log Files and Diagnostic Data in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

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

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

See Understanding the Format of DMS Events in Log Messages.

#### 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 dynamically creates a handler object that writes 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 is the logger's name followed by `-event.log`. For instance, in the example in Static Loggers and Handlers, 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:

**Table 5-9 Default locations of the logging.xml file**

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

#### 5.7.2.1.4 Using a CLI Command to Query the Trace Log File

If the logger destination's logger and handler are defined in the `logging.xml` file then you can take advantage of the `displayLogs()` command to 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` that have an ECID of `0000HpmSpLWEkJQ6ub3FEH194kwB000004`:

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

- To display only the log messages for `myTraceLogger` that 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 file name must be included:

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

## 5.7.2.2 MBean Creator Destination

**Table 5-10 MBean Creator Destination Details**

| Properties | Details |
|---|---|
| **Description** | The MBean creator destination make nouns accessible as MBeans, exposing their metrics as attributes, for access through WLDF, JConsole, and so on. |
| **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 creates 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 types that are of interest to the administrator.

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

WebLogic Diagnostic Framework (WLDF) can be used to harvest DMS metrics exposed by the MBean creator destination. See *Oracle Fusion Middleware Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

- Metric MBean Object Name

### 5.7.2.2.1 Metric MBean Object Name

The noun name and noun type are exposed as the name and type properties of the metric MBean object name. The MBean domain name is `oracle.dms`. The object name also reflects the DMS noun hierarchy.

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

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

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

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

## 5.7.2.3 Request Tracker Destination

**Table 5-11 Request Tracker Destination Details**

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

**Table 5-11 (Cont.) Request Tracker Destination Details**

| Properties | Details |
|---|---|
| **Implementing Class** | `oracle.dms.event.RequestTrackerDestination` |
| **Properties** | |
| `excludeHeaderNames` | Comma-separated list of header names to exclude from tracking. |

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

For each request the following information is dumped:

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

When the request tracker is not enabled the Request Dump outputs the following:

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

- Executing the Request Tracker Dump

### 5.7.2.3.1 Executing the Request Tracker Dump

The information maintained by the request tracker can be accessed manually. When connected to a server, to execute the dump that reports the request information the WLST `executeDump` command can be used, as follows:

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

StartTime: 2009-12-14 02:24:41.870
ECID: 0000IMChyqEC8xT6uBf9EH1B9X9^000009,0
URI: /myApp/Welcome.jsp
QueryString:
Headers:
   Host: myHost.myDomain.com:7001
   Connection: keep-alive
   User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.5
(KHTML, like Gecko) Chrome/4.0.249.30 Safari/532.5
   Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
   Accept-Encoding: gzip,deflate
   Cookie: ORA_MOS_LOCALE=en%7CGB; s_nr...
   Accept-Language: en-GB,en-US;q=0.8,en;q=0.6
   Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

## 5.7.2.4 Java Flight Recorder Destination

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

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

1. Application level events and JVM level events can be reported as a single sequence. This avoids the need to combine such events from separate log files based only on the timestamp (which might not tick over fast enough to 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 so the JVM exits 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 is associated with a JFR `InstantEvent` event type:

- The name of the JFR event type for a noun type is the noun type name with the suffix `state`.
- The path of the JFR event type for a noun type is `dms/` followed by the `producer-name`, followed by the event type name.
- Event sensors do not contribute any values to the JFR event type.
- The values of the JFR event for a noun type are described in Table 5-12:

**Table 5-12 Values of the JFR Event for a Noun Type**

| Value Name | Description | Relational | Notes |
|---|---|---|---|
| `ECID` | The Execution Context ID (ECID) associated with the action. | Yes | |
| `RID` | The RID associated with the action. | Yes | |
| `<noun type> name` | The full path of the noun. | | This field is populated with the full path of the noun. The field name assumes that the noun_type meaningfully categorizes all objects measured by the nouns of that type. |

**Table 5-12 (Cont.) Values of the JFR Event for a Noun Type**

| Value Name | Description | Relational | Notes |
|---|---|---|---|
| `<state-sensor-name>` | The value of the state sensor. | No | Each state sensor belonging to the noun contributes one of these values to the instant event. There might 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 counting 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 is associated with a JFR DurationEvent event type in the following ways:

- The name of the JFR event type for a phase sensor belonging to a noun of a particular noun type is the **noun type** name followed by the phase sensor name.
- The path of the JFR event for a noun type is `dms/` followed by the `producer-name`, followed by the event type name.
- The values of the duration event is as mentioned (except for the sensorName value). For example, the **stop** of a phase event results in a JFR duration event being reported to JFR that contains the state information of the phase event parent noun.

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

- Noun type description is used in creation of the JFR event type.
- State and event sensor descriptions are not applied–there is nowhere to apply them.
- Phase sensor descriptions are applied to their JFR event type.
- Examples of Dynamically Derived Producers and Events

### 5.7.2.4.1.1 Examples of Dynamically Derived Producers and Events

Table 5-13 provides examples for the rules described in Dynamically Derived JFR Event Types – Names, Values, and Descriptions:

**Table 5-13 Examples of Dynamically Derived Producers and Events**

| DMS | Java Flight Recorder (JFR) |
|---|---|
| **Noun type:**<br>`JDBC_Connection`<br>**Noun path:**<br>`/JDBC/Driver/CONNECTION_7`<br>**Sensors:**<br>`CreateStatement (P)`<br>`CreateNewStatement (P)`<br>`DBWaitTime (P)`<br>`JDBC_Connection_Url (S)`<br>`JDBC_Connection_Username (S)`<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 is the full path of the noun<br>• `JDBC_Connection_Url`<br>Value of the state sensor of this name at the time of the event<br>• `JDBC_Connection_Username`<br>Value of the state sensor of this name at the time of the event<br>• `Event Name`<br>Value is 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 5-13 (Cont.) Examples of Dynamically Derived Producers and Events**

| DMS | Java 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` |
| - | Producer Name: JDBC<br>**Event Type 4**<br>Event Type Name: `JDBC_Connection DBWaitTime`<br>Event Type Path:<br>`dms/JDBC/JDBC_Connection_DBWaitTime`<br>**Fields:**<br>• `ECID`<br>• `RID`<br>• `JDBC_Connection name`<br>• `JDBC_Connection_Url`<br>• `JDBC_Connection_Username` |

## 5.7.3 Understanding the Format of DMS Events in Log Messages

Table 5-14 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 5-14 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:_REQUEST:STOP:/MyWebApp/emp` |
| All | 2 | Event time | The time at which the event occurred.<br>For example:<br>`v1:**1280737384058**:_REQUEST:STOP:/MyWebApp/emp` |

**Table 5-14 (Cont.) Event Formatting Descriptions**

| Applicable Events | Field Number | Name | Description |
|---|---|---|---|
| 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>• _REQUEST<br>For example:<br>v1:1280737384058:**_REQUEST**:STOP:/MyWebApp/emp |
| All | 4 | Action type | The type of action that resulted in the generation of this event. A given source object type might not produce events for every action type:<br>• CREATE<br>• UPDATE<br>• DELETE<br>• START<br>• STOP<br>• ABORT<br>For example:<br>v1:1280737384058:_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 5-14 (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 5-14 (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`** |
| Requests | 5 | URI | Uniform Resource Identifier (URI) identifies the resource upon which to apply the request.<br>For example:<br>`v1:1280737382889:_REQUEST:START:`**`/myWebApp/showEmployees`**<br>`v1:1280737384058:_REQUEST:STOP:`**`/myWebApp/showEmployees`** |

**Table 5-14 (Cont.) Event Formatting Descriptions**

| Applicable Events | Field Number | Name | Description |
|---|---|---|---|
| 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 might contain **:** but do not interpret them 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 5-15 shows the action types that can be performed on source object types.

**Table 5-15 Actions Performed on Source Object Types**

| Object Type | 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 | - | - | - |
| Execution Context | - | - | - | Yes | Yes | - |
| Request | - | - | - | Yes | Yes | - |

# 5.8 DMS Best Practices

Implement the following best practices when you use DMS metrics.

The use of DMS metrics can have an impact on application performance. When you add 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. Use a high resolution clock to time phase events accurately than 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 might not tick (change) once per millisecond. Instead, it might take up to 15 ms to tick as shown in the following example:

**Table 5-16 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 5-16 shows a phase with a 12 ms 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 2 ms duration running from 12:00:00.014 to 12:00:00.016 would be reported in system time as having a duration of 15 ms.

> **Note:**
>
> These behaviors are more evident on some operating systems than others. Use caution when you analyze individual periods of time that are shorter than the tick period of the system clock. Configuring DMS to use a higher resolution clock causes 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 by 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 you use the high resolution clock, then set the units property as follows:
>
> `-Doracle.dms.clock.units=msecs`

Table 5-17 shows the supported values for the `oracle.dms.clock` property.

**Table 5-17 The 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 5-18 shows the supported values for the `oracle.dms.clock.units` property.

**Table 5-18 oracle.dms.clock.units Property Values**

| Value | Description |
|---|---|
| `MSECS` | Specifies that the time must 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 must be converted to microseconds and reported as `usecs`. A microsecond is $10^{-6}$ seconds. |
| `NSECS` | Specifies that the time must 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 you use 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, and HighRes.
- DMS checks the property values at startup. When the clock property is set with a value that is not listed in Table 5-17, 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 5-18, DMS uses the default units for the specified clock.

# Part II
# Core Components

The core components in Oracle Fusion Middleware need to be tuned for optimal performance.

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

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

- Tuning Oracle HTTP Server
  You can tune Oracle HTTP Server (OHS) to optimize its performance as the web server component for Oracle Fusion Middleware.
- Tuning Oracle Metadata Service
  You can tune Oracle Metadata Services (MDS) to optimize its performance as an application server and Oracle relational database.
- Tuning Oracle Fusion Middleware Security
  You can tune Oracle Fusion Middleware security services to optimize the performance of security services through Oracle Platform Security Services (OPSS) and Oracle Web Services.

# 6
# Tuning Oracle HTTP Server

You can tune Oracle HTTP Server (OHS) to optimize its performance as the web server component for Oracle Fusion Middleware.

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

- About Oracle HTTP Server
  Oracle HTTP Server (OHS) is the Web server component for Oracle Fusion Middleware.
- Monitoring Oracle HTTP Server Performance
  Oracle Fusion Middleware automatically and continuously measures runtime performance for Oracle HTTP Server.
- Basic Tuning Considerations
  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.
- Advanced Tuning Considerations
  Advanced tuning recommendations may or may not apply to your environment. Review the following recommendations to determine if the changes would improve your Oracle HTTP Server performance.

## 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 2.4.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 on the Apache open-source software infrastructure, see the Apache Software Foundation at `http://www.apache.org/`.

## 6.2 Monitoring Oracle HTTP Server Performance

Oracle Fusion Middleware automatically and continuously measures runtime 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. See Managing and Monitoring Server Processes in *Oracle Fusion Middleware Administering Oracle HTTP Server*.

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

See Viewing Metrics with WLST (Oracle WebLogic Server).

# 6.3 Basic Tuning Considerations

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

See the Oracle HTTP Server, see Understanding Oracle HTTP Server Management Tools in *Oracle Fusion Middleware Administering Oracle HTTP Server*.

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

- Worker: It uses Multi-Process-Multi-Threads model and is the default MPM on all platforms other than Microsoft Windows platforms. Multithread support makes it more scalable by using fewer system resources and multiprocess 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. It is considered the least efficient MPM.
- Event: This MPM is designed to allow more requests to be served simultaneously by passing off some processing work to supporting threads, freeing up the main threads to work on new requests. It is based on the worker MPM, which implements a hybrid multiprocess multithreaded server.

The directives for each MPM type are defined in the `ORACLE_INSTANCE/config/fmwconfig/components/OHS/<componentName>/httpd.conf`. The default MPM type is the event MPM. To use a different MPM (such as prefork MPM), edit the `ohs.plugins.nodemanager.properties` file.

> **Note:**
>
> The information here is based on the use of worker and WinNT MPMs, which use threads. The directives listed might 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. 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 |

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

| Directive | Description |
|---|---|
| `MaxRequestWorkers`<br>This directive maps to the **Maximum Requests** field on the Performance Directives screen.<br>This parameter is not available in **mod_winnt** (Microsoft Windows). Winnt uses a single process, multithreaded model and is controlled by the `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 `MaxRequestWorkers` 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 `MaxRequestWorkers` to increase server concurrency. Increase `MaxRequestWorkers` 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 `MaxRequestWorkers` 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 might require sufficient concurrent httpd or thread server processes to handle both active and idle connections. When you specify `MaxRequestWorkers` 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, during the connection, even if no requests are currently being processed for the connection.<br>If you have sufficient capacity, `KeepAlive` must be enabled; using persistent connections improves performance and prevents wasting CPU resources reestablishing connections. Normally, you should not change `KeepAlive` parameters.<br>The maximum allowed value for `MaxRequestWorkers` is 8192 (8K).<br>Default Value: 150 |

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

| Directive | Description |
|---|---|
| `StartServers`<br>This directive maps to the **Initial Child Server Processes** field on the Performance Directives screen. | Specifies the number of child server processes that are created on startup. If you expect a sudden load after restart, set this value based on the number of child servers that are required.<br>The following parameters are inter-related and applicable only on UNIX platforms (worker_mpm):<br>• `MaxRequestWorkers`<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>Each child process has a set of child threads that are defined for them and that can actually handle the requests. Use `ThreadsPerChild` with this directive.<br>The values of `ThreadLimit`, `ServerLimit`, and `MaxRequestWorkers` can indirectly affect this value. Read the notes for these directives and use them with this directive.<br>Default Value: 2 |
| `ServerLimit`<br>This parameter is not available in **mod_winnt** (Microsoft Windows). Winnt uses a single process, multithreaded 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. It is used to control the maximum number of server processes that can be created.<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. It is used to control the maximum number of threads created per process to avoid conflicts or issues.<br>Default Values:<br>• Windows Multi-Processing Module (mpm_winnt): 1920<br>• All others: 64 |

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

| Directive | Description |
|---|---|
| `ThreadsPerChild`<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>Default Value: 64 when **mpm_winnt** is used and 25 when worker MPM is used.<br>The `ThreadsPerChild` directive works with other directives, as follows:<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 or worker), as specified in `ThreadsPerChild`, and a listener thread, which listens for requests and transfers the control to the worker or server threads.<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>The `ServerLimit`, `ThreadLimit`, and `MaxRequestWorkers` directives affect the other directives as follows:<br>• `ServerLimit`: Defines the upper limit on the number of servers that can be created. This affects `MaxRequestWorkers` and `StartServers`.<br>• `ThreadLimit`: Defines the upper limit on `ThreadsPerChild`. If `ThreadsPerChild` is greater than `ThreadLimit`, then it is automatically trimmed to the latter value.<br>• `MaxRequestWorkers`: Defines the upper limit on the number of server threads that can process requests simultaneously. This must be equal to the number of simultaneous connections that can be made. This value must be a multiple of `ThreadsPerChild`. If `MaxRequestWorkers` is greater than `ServerLimit` multiplied by `ThreadsPerChild`, it is automatically 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 that 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, it 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**: Windows systems `MaxRequestsPerChild` must 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>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 more server threads to handle transient load increases such as multiple simultaneous requests from a single browser.<br>The server periodically checks how many server threads are waiting for a request. If there is fewer than `MinSpareThreads`, it creates a new spare. If there is 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 to wait for an incoming request to be received before sending a time-out.<br>Default Value: 300 |
| `KeepAlive`<br>This directive maps to the **Multiple Requests Per Connection** field on the Performance Directives screen. | Whether to allow persistent connections (more than one request per connection). Set to Off to deactivate.<br>Default Value: On |
| `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 |

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

| Directive | Description |
|---|---|
| `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 then adjusts the soft limit (`limit`) to match.<br>Configuring file descriptor limits is platform-specific. Refer to your operating system documentation for details. |

## 6.3.2 Reducing Process Availability with Persistent Connections

If your browser supports persistent connections, you can support them on the server by 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. Consider the size and behavior of your own user population when you set 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.

The `KeepAlive` option should be used judiciously along with `MaxClients` directive. The `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 the user population consistently higher than the specified `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 the 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 the `KeepAlive` connections to avoid starvation. This can impact overall performance when the user concurrency increases. System performance is impacted by increased concurrency and can possibly cause the system to fail.

`MaxClients` must 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 off the `KeepAlive` option or reduce it to a 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 you consider the nature of the workload and the system capacity.

## 6.3.3 Logging Options for Oracle HTTP Server

The logging options for Oracle HTTP Server include types of logging, log levels, and the performance implications for using logging.

- Access Logging
- Configuring the HostNameLookups Directive
- Error 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 that is 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 offline, with the `logresolve` utility found in the `/Apache/Apache/bin/` directory.

### 6.3.3.3 Error logging

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

The `d.conf` file contains the error log configuration for OHS. The `OraLogMode` directive defines the logging mode. The default is `odl-text`, which produces the Oracle diagnostic logging format in a text file. Alternatively, change it 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 might also fill disk space. The log level control must be used based on need. For requests that use dynamic resources, like `mod_osso` or `mod_plsql`, there is a performance cost associated.

## 6.4 Advanced Tuning Considerations

Advanced tuning recommendations may or may not apply to your environment. Review the following recommendations to determine if the changes would improve your Oracle HTTP Server performance.

- Tuning Oracle HTTP Server
- Tuning Oracle HTTP Server Security

### 6.4.1 Tuning Oracle HTTP Server

You can follow the topics to avoid or debug potential Oracle HTTP Server performance problems.

- Analyzing Static Versus Dynamic Requests
- Limiting the Number of Enabled Modules
- Tuning the File Descriptor Limit

### 6.4.1.1 Analyzing Static Versus Dynamic Requests

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

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

### 6.4.1.2 Limiting the Number of Enabled Modules

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

The new architecture, Oracle HTTP Server invokes the service function of each module that is loaded (in the order of definition in the `d.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 `d.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. For 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. For example, **mod_security**, when enabled, has a cost of about 10% on CPU Cost per Transaction for the specweb benchmark. Again, enable only those modules that are required by your deployed applications to save CPU time.

### 6.4.1.3 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. You can tune the file descriptor limit by configuring the operating system's hard limit for the user who starts the OHS. Once configured, the OHS adjusts the soft limit to match the operating system limit.

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

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

> **Note:**
>
> Note that this limit must be reconfigured after you apply a patch set.

## 6.4.2 Tuning Oracle HTTP Server Security

Tuning Oracle HTTP Server includes tuning the SSL and Port Tunneling.

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

### 6.4.2.1 Tuning Oracle HTTP Server Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is a protocol developed by Netscape Communications Corporation that provides authentication and encrypted communication over the Internet. Conceptually, SSL resides between the application layer and the transport layer on the protocol stack. While SSL is technically an application-independent protocol, it has become a standard for providing security over and all major web browsers support SSL.

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

- Caching SSL on Oracle HTTP Server
- Using SSL Application Level Data Encryption
- Tuning SSL Performance

#### 6.4.2.1.1 Caching SSL on Oracle HTTP Server

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

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

The `SSLSessionCacheTimeout` directive in the `ssl.conf` file determines how long the server keeps a saved SSL session (the default is 300 seconds). The 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
```

Multiple Oracle HTTP Server processes can use a saved session cache file.

Saving the 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 the saved SSL session state has some performance costs. When the 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 you use persistent connections. Oracle HTTP Server uses persistent connections by default, assuming they are supported on the client-side. In over SSL as implemented by Oracle HTTP Server, the SSL session state is kept in memory while the associated connection is persisted, a process which essentially eliminates the performance impacts that are associated with SSL session reuse (conceptually, the SSL connection is kept open along with the connection). For more information, see Reducing Process Availability with Persistent Connections.

#### 6.4.2.1.2 Using SSL Application Level Data Encryption

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

Oracle HTTP Server negotiates a cipher suite with a client based on the **SSLCipherSuite** attribute specified in the `ssl.conf` file. OHS 11*g* uses the 128 bit Encryption algorithm by default and no longer supports lower encryption.

> **Note:**
>
> The previous release [10.1.3x] used 64 bit encryption for Windows. For UNIX, the 10.x releases used the 128 bit encryption by default.

#### 6.4.2.1.3 Tuning SSL Performance

The following recommendations can assist you to determine performance requirements when you work with Oracle HTTP Server and SSL.

- The SSL handshake is a 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 the 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. If the application requires server authentication only, do not use client authentication. If additional hardware is required, the performance goals of an application cannot be met by this method.
- Design the application to use SSL efficiently. Group secure operations to take advantage of SSL session reuse and SSL connection reuse.
- Use persistent connections, if possible, to minimize the cost of SSL session reuse.

- Tune the session cache timeout value (the `SSLSessionCacheTimeout` directive in the `ssl.conf`) file. 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, must be completed without incurring session creation more than once. The default value for the `SSLSessionCacheTimeout` attribute is 300 seconds. Test the application usability to help tune this setting.
- If large volumes of data are being protected through SSL, pay close attention to the cipher suite being used. The `SSLCipherSuite` directive specified in the `ssl.conf` file controls the cipher suite. If lower levels of security are acceptable, use a less-secure protocol by using a smaller key size (improves performance significantly). Finally, test the application by using each available cipher suite for the specified security level to find the optimal suite.
- If SSL remains a bottleneck to the performance and scalability of your application, after taking the preceding considerations into account, consider deploying multiple Oracle HTTP Server instances over a hardware cluster or consider the use of SSL accelerator cards.

### 6.4.2.2 Tuning Oracle HTTP Server Port Tunneling

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

Except for 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; it 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. Ensure to check the number of open ports. Start processes equivalent to the number of open ports.

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 must be compared with a system without OracleAS Port Tunneling to determine whether your performance requirements can be met by using OracleAS Port Tunneling.

7

# Tuning Oracle Metadata Service

You can tune Oracle Metadata Services (MDS) to optimize its performance as an application server and Oracle relational database.

- About Oracle Metadata Services (MDS)
  Oracle Metadata Services (MDS) is an application server and Oracle relational database that keeps metadata in these areas: the `ClassPath`, the `ServletContext`, database repository and, sometimes, the file system.
- Monitoring Oracle Metadata Service Performance
  MDS uses DMS sensors to provide tuning and diagnostic information, which can be viewed by using Enterprise Manager. This information is useful, for example, to see if the MDS caches are large enough.
- Basic Tuning Considerations
  Tuning the MDS configuration is essential for improving performance.
- Advanced Tuning Considerations
  After you have performed recommended modifications, you can make additional changes that are specific to your deployment. Consider carefully whether the advance tuning recommendations are appropriate for your environment.

## 7.1 About Oracle Metadata Services (MDS)

Oracle Metadata Services (MDS) is an application server and Oracle relational database that keeps metadata in these areas: the `ClassPath`, the `ServletContext`, database repository and, sometimes, the file system.

One of the primary uses of MDS is to store customizations and persisted personalization for Oracle applications. MDS is used by components such as Oracle Application Development Framework (ADF) to manage metadata. Examples of metadata objects managed by MDS are: JSP pages and page fragments, ADF page definitions and task flows, and customized variants of those objects.

> **Note:**
>
> Most of the Oracle Metadata Services configuration parameters are immutable and cannot be changed at runtime unless otherwise specified.

Tuning MDS tablespace and cache size is important before you tune Oracle B2B and other Oracle products. If you are using the *Oracle Fusion Middleware User's Guide for Oracle B2B* to tune B2B, make sure you have completed the tuning described here first.

# 7.2 Monitoring Oracle Metadata Service Performance

MDS uses DMS sensors to provide tuning and diagnostic information, which can be viewed by 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 the MDS configuration is essential for improving performance.

The default MDS configuration must be tuned in almost all deployments. It is important to review the requirements and recommendations carefully.

- Tuning Database Repository
- Tuning Cache Configuration
- Purging Document Version History
- Using Database Polling Interval for Change Detection

## 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.

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

- Collecting Schema Statistics
- Increasing Redo Log Size
- Reclaiming Disk Space
- Monitoring the Database Performance

### 7.3.1.1 Collecting Schema Statistics

While MDS provides database indexes, they might 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 the 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 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 must 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 that is consumed by MDS schema to grow. 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.

See:

- Generating Automatic Workload Repository Reports in *Oracle Database Performance Tuning Guide*.
- Monitoring Performance in *Oracle Database Performance Tuning 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 might 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 might 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 Administering Oracle Fusion Middleware*.

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

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. Manually update the cache-config in the `adf-config.xml` file by adding the following entry prior to packaging the Enterprise ARchive (EAR) file:

```
<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.

- Enabling Document Cache

### 7.3.2.1 Enabling Document Cache

In addition to the main MDS cache, MDS uses a document cache 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 small value, MDS uses a minimum size of 500 for document cache.

In general, the defaults must be sufficient usually. However, insufficient document cache size might impact performance. Set document cache size by adding this entry to the `adf-config.xml` file prior to packaging the Enterprise ARchive (EAR) file:

```
<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 Monitoring Oracle Metadata Service Performance ) must 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, you can purge version history automatically or manually.

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

- Using Auto Purge
- Purging Manually

### 7.3.3.1 Using 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 the `adf-config.xml` file prior to packaging the EAR:

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

In the example above, the auto-purge is executed every *T* seconds and 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 Administering Oracle Fusion Middleware*.

#### 7.3.3.2 Purging Manually

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

See Purging Metadata Version History in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

### 7.3.4 Using Database Polling Interval for Change Detection

MDS employs a polling thread, which queries the database to check if the data in the MDS in-memory cache is out of sync with data in the database. It happens 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. Configure the polling interval by adding this entry in the `adf-config.xml` file prior to packaging the Enterprise ARchive (EAR) file:

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

In the example mentioned, *`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 is suitable for most purposes. See Changing MDS Configuration Attributes for Deployed Applications in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

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

## 7.4 Advanced Tuning Considerations

After you have performed recommended modifications, you can make additional changes that are specific to your deployment. Consider carefully whether the advance tuning recommendations are appropriate for your environment.

## 7.4.1 Analyzing Performance Impact from Customization

MDS customization might 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 that 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: 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): are layers of customization corresponding to customization classes whose `getCacheHint` method returns `SINGLE_USER`, meaning the layer applies to one user. User customizations are cached on the user's session (Session) until the user logs out.

For details on customization concepts, writing customization classes, and configuring customization classes, see Customizing Applications with MDS in *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework*.

# 8
# Tuning Oracle Fusion Middleware Security

You can tune Oracle Fusion Middleware security services to optimize the performance of security services through Oracle Platform Security Services (OPSS) and Oracle Web Services.



## 8.1 About Security Services

Oracle Fusion Middleware provides security services through Oracle Platform Security Services (OPSS) and Oracle Web Services.

- Oracle Platform Security Services

  Oracle Platform Security 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:

  1. Configure and invoke a login module, as appropriate. You can use provided login modules, or you can use custom login modules.
  2. Authenticate the user attempting to log in, which is the role of the identity store service.
  3. Authorize the user by checking permissions for that role.
- Oracle Web Services Security

  Oracle Web Services Security provides a framework of authorization and authentication for interacting with a web service by using XML-based messages.

> **Note:**
>
> The information here assumes that you have reviewed and understand the concepts and administration information for Oracle Fusion Middleware Security Services. See, *Oracle Fusion Middleware Administering Web Services* before you tune any security parameters.

## 8.2 Basic Tuning Considerations

Tuning considerations might improve the performance of the Oracle Fusion Middleware security services.

If you discover a performance bottleneck, you must 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, 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 might need to change how you manage your database or LDAP connections or how you secure the resources.

## 8.3 Tuning Oracle Platform Security Services

Oracle Platform Security Services (OPSS) includes the following basic tuning configurations.

- JVM Tuning Parameters
- JDK Tuning Parameters
- Authentication Tuning Parameters
- Authorization Tuning Properties
- OPSS PDP Service Tuning Parameters

## 8.3.1 JVM Tuning Parameters

Tuning the JVM parameters can greatly improve performance. For example, the JVM Heap size should be tuned depending upon the number of roles and permissions in the store. At runtime, all roles and permissions are stored in the in-memory cache. For more JVM tuning information, see Tuning Java Virtual Machines (JVMs).

## 8.3.2 JDK Tuning Parameters

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

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

See the JDK 7 release notes at `http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html`

## 8.3.3 Authentication Tuning Parameters

For OPSS Authentication tuning, see "Improving the Performance of WebLogic and LDAP Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* guide on the Oracle Technology Network `http://download.oracle.com/docs/cd/E12840_01/wls/docs103/secmanage/atn.html#wp1199087`.

## 8.3.4 Authorization Tuning Properties

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

**Table 8-1 Authorization Properties**

| Java System Properties | Default Value | Valid Values | Notes |
|---|---|---|---|
| `-Djps.subject.cache.key` | `4` | 3<br>4<br>5 | JPS uses a Subject Resolver to `convert` a platform subject to a JpsSubject, which contains the `user/enterprise-role` information, as well as the `ApplicationRole` information. This information is represented as principals in the subject.<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` the Platform subject and the JpsSubject. Two subjects could be confusing when their contents are the same, but the case of the principals' name is different.<br>The following settings can be used to configure the cache key:<br>• 3: Use the platform subject directly as the key. **Note:** On WLS, if the `principalEqualCaseinsensitive` flag is enabled, two subjects could be confusing if their contents are the same, but the case of the principals is different.<br>• 4: This setting is similar to **3**but overcomes the case-sensitive issue. This is the ready—to—use setting.<br>• 5: Instead of using the whole subject as the key, this setting uses a subset of the principal set inside the subject as the key (actually use principals of the `WLSUSerImpl` type).<br>This setting accelerates the cache |

**Table 8-1 (Cont.) Authorization Properties**

| Java System Properties | Default Value | Valid Values | Notes |
|---|---|---|---|
| | | | 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. |
| `-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 has expired, the cached value is dumped. The setting is controlled by the `-Djps.subject.cache.ttl=xxxx`flag, where **xxx**is the duration in milliseconds.<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 the flag `-Djps.combiner.optimize=true` can improve the 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 the flag `-Djps.combiner.optimize.lazyeval=true` can improve the Java authorization performance. |

**Table 8-1 (Cont.) Authorization Properties**

| Java System Properties | Default Value | Valid Values | Notes |
|---|---|---|---|
| `-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>The OPSS Java Policy Provider reads from both `java.policy` and `system-jazn-data.xml`. When starting the Weblogic server, the **Hybrid** mode can be disabled by setting the system property `jps.policystore.hybrid.mode` to `false` . Setting `-Djps.policystore.hybrid.mode=false` can reduce the runtime overhead. |
| `-Djps.authz=ACC` | `ACC` | `ACC`<br>`SM` | Delegates the call to JDK API `AccessController.checkPermission`, which can reduce the performance impact at runtime or while debugging.<br>ACC: Delegate the call to `AccessController.checkPermission`.<br>SM: If SecurityManager is set, delegate the call to `SecurityManager` . |

## 8.3.5 OPSS PDP Service Tuning Parameters

Table 8-2 describes OPSS tuning parameters for policy store:

**Table 8-2 OPSS PDP Service Tuning Parameters**

| Parameter | Default Value | Valid Values | Notes |
|---|---|---|---|
| `oracle.security.jps.policystore.rolemember.cache.type` | `STATIC` | `STATIC`, `SOFT`, `WEAK` | 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 to 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 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 small.<br>Consider maintaining the default value for best performance. |

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

| Parameter | Default Value | Valid Values | Notes |
|---|---|---|---|
| `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. |
| `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 takes 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 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. |

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

| Parameter | Default Value | Valid Values | Notes |
|---|---|---|---|
| `oracle.security.jps.policystore.cache.updatable` | `True` | `True`<br>`False` | This property is used to enable refresh. Consider maintaining the default value for the best performance. |
| `oracle.security.jps.policystore.refresh.enable` | `True` | `True`<br>`False` | This property is used to enable refresh. 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 are higher than the number of application roles set to **True**; set to False otherwise, that is, when the number of application roles are high. |

# 8.4 Oracle Web Services Security Tuning

Oracle Web Services Security provides a framework of authorization and authentication for interacting with a web service by using XML-based messages. There are several factors that may affect performance of the web service.

- Choosing the Right Policy
- Policy Manager
- Configuring the Log Assertion to Record SOAP Messages

- Configuring Connection Pooling
- Monitoring the Performance of Web Services

## 8.4.1 Choosing the Right Policy

Oracle Web Services Security supports many policies and the appropriate policies must be implemented based on the security need of the deployment. Careful consideration should be given to performance, since each additional policy can impact performance. For example, Transport-level security (SSL) is faster than Application-level security, but Transport-level security can be vulnerable in multistep transactions. Application-level security has more performance implications, but provides end-to-end security.

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

## 8.4.2 Policy Manager

There is an inherent performance impact when you use the database-based policy enforcement. When database policy enforcement is chosen, careful consideration must be given to the **polling**frequency of the agent to the database.

## 8.4.3 Configuring the Log Assertion to Record SOAP Messages

The request and response pipelines of the default policy include a log assertion that causes policy enforcement points (PEP) to record SOAP messages to either a database or a component-specific local file. There can be potential performance impacts to the logging level. To prevent performance issues, consider using the lowest logging level that is appropriate for your deployment.

The following logging levels can be configured in the log step:

- Header: Only the SOAP header is recorded.
- Body: Only the message content (body) is recorded.
- Envelope: The entire SOAP envelope, which includes both the header and the body, is recorded. Any attachments are not recorded.
- All: The full message is recorded. It includes the SOAP header, the body, and all attachments, probably the URLs existing outside the SOAP message itself.

> **Note:**
>
> Typically, system performance improves when log files are located in topological proximity to the enforcement component. If possible, use multiple distributed logs in a highly distributed environment.

## 8.4.4 Configuring Connection Pooling

When you request that a Context instance use connection pooling by using the `com.sun.jndi.ldap.connect.pool` environment property, the connection that is used

might or might not be pooled. The default rule is that plain (non-SSL) connections that use simple or no authentication are allowed to be pooled. You can change this default to include SSL connections and the DIGEST-MD5 authentication type by using system properties. To allow both plain and SSL connections to be pooled, set the `com.sun.jndi.ldap.connect.pool.protocol` system property to the string `plain ssl` as shown below:

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

## 8.4.5 Monitoring the Performance of Web Services

You can monitor the performance on the following Oracle Web Services through the Web Services home page of Oracle Fusion Middleware Control:

- Endpoint Enabled Metrics such as:
  - Policy Reference Status
  - Total Violations
  - Security Violations
- Invocations Completed
- Response Time, in seconds
- Policy Violations such as:
  - Total Violations
  - Authentication Violations
  - Authorization Violations
  - Confidentiality Violations
  - Integrity Violations
- Total Faults

For general information on monitoring Oracle Fusion Middleware components, see .

For detailed information on using Oracle Fusion Middleware Control to monitor Oracle Web Services, see Overview of Performance Monitoring, Auditing, and Tuning in *Oracle Fusion Middleware Administering Web Services*.

# Part III

# Oracle Fusion Middleware Server Components

The Oracle Fusion Middleware server components need to be tuned for optimal performance.

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

- Tuning Oracle Application Development Framework (ADF)
  You can tune Oracle Application Development Framework (ADF) to optimize its performance and scalability with design, configuration, and deployment considerations.
- Tuning Oracle TopLink
  You can tune EclipseLink, an open-source persistence framework used with Oracle TopLink, to optimize its performance as the Java Persistence API (JPA) implementation.

9

# Tuning Oracle Application Development Framework (ADF)

You can tune Oracle Application Development Framework (ADF) to optimize its performance and scalability with design, configuration, and deployment considerations.

> **Note:**
>
> - *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework*
> - *Oracle Fusion Middleware Developing Web User Interfaces with Oracle ADF Faces*



## 9.1 About Oracle ADF

Oracle Application Development Framework (Oracle ADF) is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications.

Oracle ADF is suitable for enterprise developers who want to create applications that search, display, create, modify, and validate data by using web, wireless, desktop, or web services interfaces. If you develop enterprise solutions that search, display, create, modify, and validate data by 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 Developing Fusion Web Applications with Oracle Application Development Framework*.

# 9.2 Basic Tuning Considerations

To achieve optimal performance, you can follow tuning recommendations before you build, configure, and deploy ADF applications.

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

## 9.2.1 Oracle ADF Faces Configuration and Profiling

Configuration options for Oracle ADF Faces are set in the `web.xml` file. Most of these options have default values that are tuned for performance. Table 9-1 describes some of these configuration options.

**Table 9-1 ADF Configuration Options**

| Parameter | Description |
|---|---|
| Compression View State<br>`org.apache.myfaces.trinidad.COMPRESS_VIEW_STATE` | Controls whether the page state is compressed. If the size of the data is compressed, latency can be reduced. This parameter should be set to `True`. |
| Enhanced Debug<br>`org.apache.myfaces.trinidad.resource.DEBUG` | Controls whether output should be enhanced for debugging. This parameter should be removed or set to `False`. |
| Check File Modification<br>`oracle.adf.view.rich.CHECK_FILE_MODIFICATION` | Controls whether ADF faces check for modification date of JSP pages and discards any saved state if the file is changed. This parameter should be removed or set to `False`. |
| Client State Method<br>`oracle.adf.view.rich.CLIENT_STATE_METHOD` | Specifies the type of saving (`all` or `token`) that should be used when client-side state saving is enabled. The default value is `token`. |
| Client-Side Log Level<br>`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`. |
| Assertion Processing<br>`oracle.adf.view.rich.ASSERT_ENABLED` | Specifies when 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 by time using the Firefox browser, ensure that the Firebug plug-in is disabled. While this plug-in is very useful for getting information about the page and for debugging JavaScript code on the page, it can impact the total response time.
>
> For more information on disabling the Firefox Firebug plug-in, see the Firefox Support Home Page at `http://support.mozilla.com/en-US/kb/`.

## 9.2.2 Performance Considerations for ADF Faces

Table 9-2 provides configuration recommendations that may improve performance of ADF Faces:

**Table 9-2 Configuration Parameters for ADF Faces**

| Configuration Recommendation | Description |
|---|---|
| Avoid inline JavaScript in pages. | Inline JavaScript can increase response payload size, is never cached in the 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 by using `af:resource` tag.<br>**TIP**: 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>*x*<br>`</param-value>`<br>`</init-param>`<br>`</servlet-name>`<br>`</servlet>`<br>**Note:**<br>Set parameter *x* based on your own use case scenarios. |

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

| Configuration Recommendation | Description |
|---|---|
| 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 and put all the input fields inside it. This drop down 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 cuts down the unnecessary DOM or client components sent to the browser. Similarly, the DOM has a `panelGroupLayout` with a number of cells that 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. |
| 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>To prevent the performance impact, consider increasing the timeout limit to 300ms. |

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

| Configuration Recommendation | Description |
|---|---|
| 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 event for cached images immediately after the PPR content is fetched.<br>For Internet Explorer 8 use a timeout (10ms) to call `_prepareForIncompleteImages` so that the image tag called right after the `.xml` request is processed. Note that this is not an issue for Mozilla Firefox or Google Chrome. |
| 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 by using Partial Page Rendering (PPR)/Ajax channel.<br>The main advantage of partial page navigation over traditional full page navigation is improved performance: the browser no longer reinterprets and reexecutes Javascript libraries, and does not spend time for cleanup or initialization of the full page. The performance benefit from this optimization is very big; it should be enabled whenever possible.<br>Some known limitations of this feature are:<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>• Applications cannot use anchor (hash) URLs for their own purposes. |

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

| Configuration Recommendation | Description |
|---|---|
| Use page templates. | 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 on the page they are building.<br>There are some important considerations when using templates:<br>• Since templates are present in every application page, they have to be optimized so that common performance impacts are avoided. For example, adding round corners to the template, can impact the performance for every page.<br>• When building complex templates, sometimes it is easier to build them in multiple pieces and include them in the top-level template by using the `<f:subview>` tag. However, from a performance perspective, this is not typically recommended since it can impact memory usage on the server side. The `<f:subview>`tag 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 the `<f:subview>` tag unless it is required. If you can ensure that all IDs are unique, it is not necessary to use the `<f:subview>` around `<jsp:include>` . For example, if you are using `<jsp:include>`, break a large page into multiple pieces for easier editing. And whenever possible, avoid using the `<f:subview>` tag. If you are including content developed by someone else, use the `<f:subview>` tag if you do not know which IDs the developer used. In addition, you do not have to put the `<f:subview>` tag at the top of a region definition.<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 9-2 (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 you use geometry management, try to minimize 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`. The `Toolbar` component 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 9-2 (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>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>Two main Ajax patterns are implemented with partial page rendering (PPR):<br>• native component refresh<br>• cross-component refresh<br>While the framework builds in native component refresh, cross-component refresh has to be done by developers in certain cases.<br>Cross-component refresh is implemented declaratively or programmatically by the application developer defining the components that are to trigger a partial update and the other components that are to act as partial listeners, and so be updated. Using cross-component refresh and implementing it correctly is one of the best ways to improve client-side response time. While designing the UI page always think about what should happen when the user clicks a command button. Is it needed for the whole page to be refreshed or only the output text field? What should happen if the value in some field is updated? For more information, see *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework*.<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, user 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>• 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 the components that are to be refreshed when the partial submit takes place, in this example |

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

| Configuration Recommendation | Description |
|---|---|
| | 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>The steps above achieve PPR by using a command button to trigger the partial page refresh.<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>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. |
| 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>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, and 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 that all the resources are present on the server.<br>Consider using the `ResourceServlet` to configure the `web.xml` file to enable resource caching:<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>` |

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

| Configuration Recommendation | Description |
|---|---|
| Reduce the size of state token cache. | This property is defined in `web.xml` `org.apache.myfaces.trinidad.CLIENT_STATE_MAX_TOKENS` in token-based client-side state saving and determines how many tokens should be preserved at any one time. The default value is `15`. When this value is exceeded, state is forgotten for the least recently viewed pages, which can impact users that actively use the **Back** button or have multiple windows that are open simultaneously.<br>To reduce live memory per session, consider reducing this value to `2`. Reducing the state token cache to `2` means one **Back** button click is supported. For applications without support for a **Back** button, this value should be set to `1`. |

**Table 9-2 (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>An HTML page basically has two parts, the `head` and the `body`. When you add an `af:document` component to a 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>To get a component (or static CDATA content) to display in the `head`, use the `metaContainer` facet as follows:<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>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. |

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

| Configuration Recommendation | Description |
|---|---|
| | See *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework* for details about `af:facetRef`. |
| 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 a 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>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 you customize the Javascript code the 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>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>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 pop-up 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 the `trinidad-config.xml` file. |

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

| Configuration Recommendation | Description |
|---|---|
| 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 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 response headers do not tell the browser that resources (JS libraries, CSS style sheets, or images) can be cached.<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 the jsp or the jspx files are checked for modifications each time they are accessed.<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. |
| Disable checking for CSS file modifications. | The `org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION` parameter controls when the CSS file modification checks are made. To aid in performance, this configuration option defaults to `false`-does 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 the page size. In production environments, make sure that you remove the `DISABLE_CONTENT_COMPRESSION` parameter from the `web.xml` file or set it to `FALSE`.<br>For debugging, turn off the style class content compression. You can do this by setting the `DISABLE_CONTENT_COMPRESSION` property to `TRUE`. |

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

| Configuration Recommendation | Description |
|---|---|
| Enable JavaScript obfuscation. | ADF Faces supports a runtime 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>To provide an obfuscated ADF Faces build, set the `org.apache.myfaces.trinidad.DEBUG_JAVASCRIPT` parameter to `FALSE` in the `web.xml` file.<br>There are two ways to check that the code is obfuscated by using Firefox with Firebug enabled:<br>Check the download size:<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>Check the source:<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>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 the 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. |

## 9.2.3 Tuning ADF Faces Component Attributes

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

**Table 9-3 ADF Faces Component Attributes**

| Configuration Recommendation | Description |
|---|---|
| Use the `immediate` attribute. | 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 that are 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>• To avoid processing the data from the current screen while navigating to the new page set the `immediate` attribute to `TRUE`in the `commandNavigationItem` in the `navigationPane`.<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 Using the JSF Lifecycle with ADF Faces in *Oracle Fusion Middleware Developing Web User Interfaces with Oracle ADF Faces*.<br>There are some important issues that are associated with the `immediate` attribute. Refer to Using the Immediate Attribute in *Oracle Fusion Middleware Developing Web User Interfaces with Oracle ADF Faces* for more information.<br>Note that this is an advanced feature. Most of the performance improvements can be achieved by using the `af:subform` component. |
| 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 whether the component is to be displayed on the page or 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 that there is no client interaction with the component. Making a component not rendered can improve server performance and client response time since the component does not have client-side representation. |

**Table 9-3 (Cont.) ADF Faces Component Attributes**

| Configuration Recommendation | Description |
|---|---|
| Use client-side events. | ADF Rich Client framework provides the client-side event model based on component-level events rather than DOM level. The client-side event model is a very useful feature that can speed up the application. Review the following performance considerations:<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-side by the client components are propagated to the server. If a client-side event handler is provided, consider canceling the event at the end of processing so that the event does not propagate to the server. |
| Use the `id` attribute. | The `id` attribute should not be longer than 7 characters in length. This is particularly important for naming containers. A long id can impact performance as the amount of HTML that must be sent down to the client is impacted by the length of the ids. |
| Use client-side components. | ADF Rich Client framework has client-side components that play a role in client-side event handling and component behavior. The `clientComponent` attribute is used to configure when (or if) a client-side component should be generated. Setting `clientComponent` attribute to `TRUE` has a performance impact, so determine if its necessary to generate client-side components.<br>For more information, see What Happens When You Set clientComponent to true in *Oracle Fusion Middleware Developing Web User Interfaces with Oracle ADF Faces*. |
| 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 cannot 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 cannot be used if you need to refer to any child components of the popup before the popup is displayed. Setting `childCreation="deferred"` postpones creating any child components of the popup and you cannot refer to them until after the popup is shown. |

## 9.2.4 Performance Considerations for Table and Tree Components

`Table`, `Tree`, and `TreeTable` are some of the most complex, and frequently used, components. Since these components can include large sets of data, they can be the common source of performance problems. Table 9-4 provides some performance recommendations.

**Table 9-4 Table and Tree Component Configurations**

| Configuration Recommendation | Description |
|---|---|
| Use `editingMode="clickToEdit"`. | When using `editingMode="editAll"` all content of the editable values holders and their client components is sent. This can significantly increase the payload and the Document Object Model (DOM) content on the client.<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>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>For more information, see Using Tables, Trees, and Other Collection-Based Components in *Oracle Fusion Middleware Developing Web User Interfaces with Oracle ADF Faces*. |
| 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 you enable 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. |

**Table 9-4 (Cont.) Table and Tree Component Configurations**

| Configuration Recommendation | Description |
|---|---|
| Consider using `visitTree` instead of `invokeOnComponent`. | A partial visit using `visitTree` is always at least as fast as `invokeOnComponent`. In addition, for components that control visiting, providing both `invokeOnComponent` and `visitTree` implementations is a source of errors. Consider deprecating `invokeOnComponent` and use `visitTree` instead.<br>For more information, see Using Tables, Trees, and Other Collection-Based Components in *Oracle Fusion Middleware Developing Web User Interfaces with Oracle ADF Faces*. |

## 9.2.5 Performance Considerations for autoSuggest

The `autoSuggest` feature 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. To speed up database processing, a database index should be created on the column for which autosuggest is enabled. This improves the component's response times especially when the database table has a large number of rows.

## 9.2.6 Data Delivery - Lazy versus Immediate

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

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

When choosing between these two options, consider the following:

| Delivery option | Description |
|---|---|
| Lazy Delivery (default) | Lazy delivery should be used for tables, or other stamped components, which are known to have slow fetch time. For example, the 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 is shorter, and the user perceives better performance.<br>Lazy delivery is implemented by using the 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>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 mechanism to deliver data in lazy mode, parallel execution of data controls is not. To enable parallel execution, open the page definition and change `RenderHint` on the iterator to `background`.<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 requests in parallel and possibly more database connections are opened.<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 task flows). When there is a single data control frame, parallel execution may not improve performance, since parallel execution synchronizes on the data control frame level. |
| Immediate Delivery | Immediate delivery (`contentDelivery="immediate"`) should be used if the table data control is fast, or if it returns a small set of data. In these cases, the response time is faster than using lazy delivery.<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. |

## 9.2.7 Performance Considerations for DVT Components

DVT components are data visualization components built on top of ADF Rich Client components. DVT components include graphs, gauges, Gantt charts, pivot tables and maps. Table 9-5 provides some configuration recommendations for DVT components:

**Table 9-5 DVT Component Configurations**

| Configuration Recommendation | Description |
|---|---|
| Modify the `RangeSize` attribute. | The `RangeSize` attribute defines the number of rows to return simultaneously. A `RangeSize` value of `-1` causes the iterator to return all the rows. Using a lower value may improve performance, but it may be harder to stop the data and any data beyond `RangeSize` is not available in the view. |
| Use horizontal text instead of vertical text. | 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. |

# 9.3 Advanced Tuning Considerations

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

- ADF Server Performance

## 9.3.1 ADF Server Performance

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

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

- Tuning Session Timeout

- Tuning View Objects
- Enabling Batch Processing
- Tuning RangeSize
- Configuring Application Module Pooling
- Using ADFc Regions
- Deferring Task Flow Execution
- Deferring Task Flow Creation in Popups
- Configuring the Task Flow Inside Switcher
- Reusing Static Data
- Conditional Validations

### 9.3.1.1 Tuning Session Timeout

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

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

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

### 9.3.1.2 Tuning View Objects

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. *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework* provides detailed information on tuning View Objects. Provided here are some tips pertaining to View Object performance.

- Creating View Objects
- Configuring View Object Data Fetching
- Setting Additional View Object Configurations

#### 9.3.1.2.1 Creating View Objects

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

**Table 9-6 Types of View Objects**

| View Object Type | Description |
|---|---|
| Read-only View Objects | If the View Object does not have to insert or update data, consider using a read-only View Object. 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 runtime, which retrieve attributes needed in the UI, data reads from local cache (instead of reexecuting 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 that are 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 you use 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 runtime by using the `createViewObjectFromQueryStmt()` API on the AM. However, avoid using runtime-created View Objects, unless absolutely necessary, due to potential performance impacts and complexity of tuning. |

### 9.3.1.2.2 Configuring View Object Data Fetching

View Object performance is largely dependent on how the view object is configured to fetch data. If the fetch options are not tuned correctly for the application, then the view object may fetch an excessive amount of data or may take too many round-trips to the database. Fetch options can be configured through the **Retrieve from the Database** group box in the View Object dialog Figure 9-1.

**Figure 9-1 View Object Dialog**

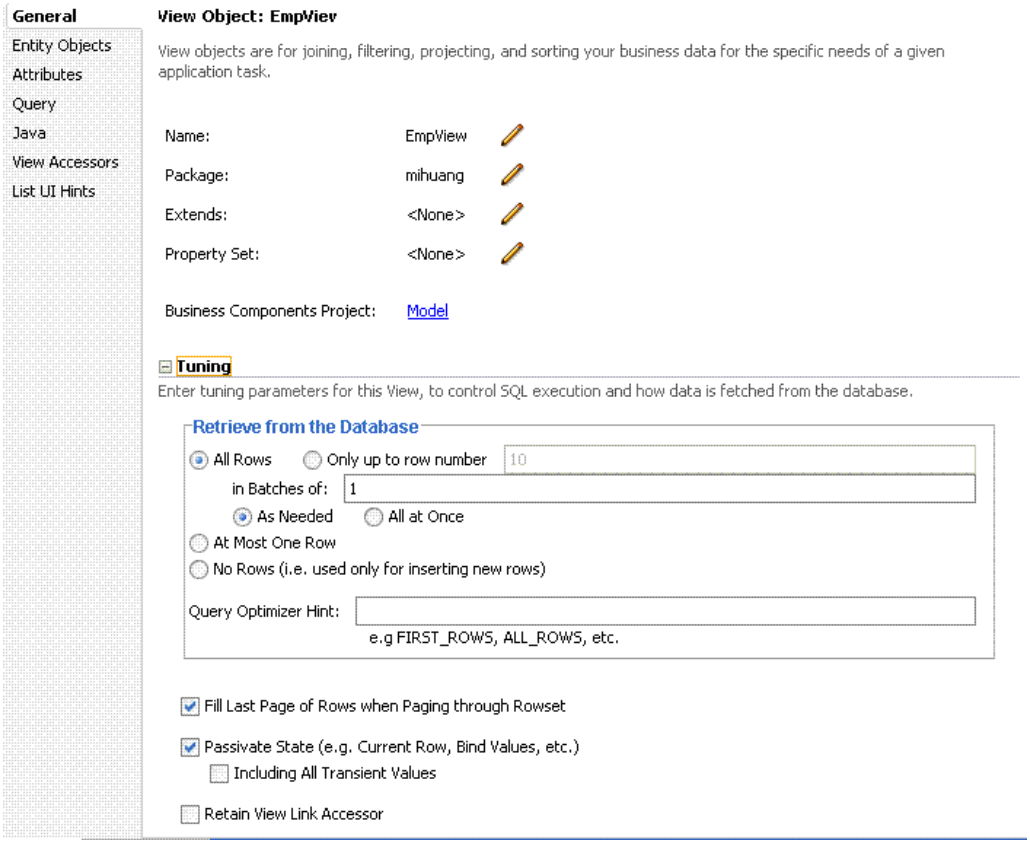



**Table 9-7 View Object Configurations**

| 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 to the *n*+1 value where *n* is the number of rows that are displayed in the user interface.<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. |

**Table 9-7 (Cont.) View Object Configurations**

| Fetch Option | Description |
|---|---|
| `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>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>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. |
| `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. |

### 9.3.1.2.3 Setting Additional View Object Configurations

Table 9-8 provides additional tuning considerations when you use the View Object:

**Table 9-8 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 you navigate 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. When defining the database connection, the SQL style is set. By default, when you define an Oracle database connection, the SQL style is `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 that is associated with the View Object contains values that may change from execution to execution, consider using bind variables. This may help to avoid reparsing 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. |

**Table 9-8 (Cont.) Additional View Object Configurations**

| Configuration Recommendation | Description |
|---|---|
| Use dynamic SQL generation. | View Objects can be configured to dynamically generate SQL statements at runtime instead of defining the SQL at design time. A View Object instance, that is configured with generating SQL statements dynamically, can avoid requerying a database. This is especially true during page navigation if a subset of all attributes with the same key Entity Object list is used in the subsequent page navigation. Performance can be improved by activating a superset of all the required attributes to eliminate a subsequent query execution. |

### 9.3.1.3 Enabling 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:**
>
> the `BatchThreshold` configuration for the EO is not compatible if an attribute in the EO exists with the configuration to refresh after insert (`RetrievedOnInsert="true"`) or update (`RetrievedOnUpdate="true"`).

### 9.3.1.4 Tuning RangeSize

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

### 9.3.1.5 Configuring Application Module Pooling

Application module (AM) pooling enables multiple users to share several application module instances. The configurations for the AM pool vary depending on the expected usage of the application.

Most of the AM pool parameters can be set through Oracle JDeveloper. The configurations are saved in the `bc4j.xcfg`file, 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 9-9 Application Module (AM) Pool Tuning**

| Configuration Recommendation | Description |
|---|---|
| Optimize the number of AM pools in the application. | Parameters that are applied at the system level are applied per AM pool. If the application uses more than 1 AM pool, then the 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.<br>For example, if an application uses four 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).<br>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 the `bc4j.xcfg`file. |
| Optimize the number of database connections. | By default, AM instances retain their database connections even when checked back into the AM pool. There are many performance benefits to maintain this association. To maintain performance, consider configuring more AM instances than the maximum number of specified database connections. |

**Note:**

If you have an AM pool that needs to be used as the 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.

- General AM Pool Configurations
- Configuring Application Module Pool Sizing
- Configuring Application Module Pool Resource Cleanup
- Designing an Application Module

### 9.3.1.5.1 General AM Pool Configurations

Use the following guidelines as a general starting point when tuning AM and AM pool behavior. More specific tuning for memory or CPU usage can be found in Configuring Application Module Pool Sizing.

**Table 9-10 AM Pool Tuning Parameters**

| Parameter | Description |
|---|---|
| Initial Pool Size<br>`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 that require an AM instance.<br>Configure this value to 10% more than the anticipated number of concurrent AM instances that are required to service all users. |
| Maximum Pool Size<br>`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 the specified limit. A general guideline is to configure this to 20% more than the initial pool size to allow for some additional growth. |
| Minimum Available Size<br>`jbo.ampool.minavailablesize` | The minimum number of available application module instances that the pool monitor should leave in the pool during a resource cleanup operation, when the server is under light load.<br>If you want the pool to shrink to contain no instances when all instances have been idle for longer than the idle time-out after a resource cleanup, set to `0` (zero).<br>The default is `5` instances.<br>While application module pool tuning allows different values for the `jbo.ampool.minavailablesize` \| `jbo.ampool.maxavailablesize` parameters, in most cases it is fine to set these minimum and maximum tuning properties to the same value. |

**Table 9-10 (Cont.) AM Pool Tuning Parameters**

| Parameter | Description |
|---|---|
| Maximum Available Size<br>`jbo.ampool.maxavailablesize` | The ideal maximum number of available application module instances in the pool when the server is under load.<br>When the pool monitor wakes up to do resource cleanup, it will try to remove available application module instances to bring the total number of available instances down to this ideal maximum. Instances that have been not been used for a period longer than the idle instance time-out always get cleaned up at this time. Then, additional available instances are removed, if necessary, to bring the number of available instances down to this size.<br>The default maximum available size is `25` instances. Configure this to leave the maximum number of available instances desired after a resource cleanup. A lower value generally results in more application module instances being removed from the pool during cleanup.<br>While application module pool tuning allows different values for the `jbo.ampool.maxavailablesize` \| `jbo.ampool.minavailablesize` parameters, in most cases it is fine to set these minimum and maximum tuning properties to the same value. |
| Referenced Pool Size<br>`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`).<br>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.<br>Configure this value to the expected number of concurrent users that perform multiple operations with short think times. If there are no users expected to use the application with short think times, then this can be configured to 0 (zero) to eliminate affinity. |
| Maximum Instance Time to Live<br>`jbo.ampool.timetolive` | The number of milliseconds after which to consider a connection instance in the pool as a candidate for removal during the next resource cleanup, regardless of whether it would bring the number of instances in the pool below minimum available size.<br>The default is `3600000` milliseconds of total time to live (which is 3600 seconds, or one hour). A lower value reduces the time an application module instance can exist before it must be removed at the next resource cleanup. The default value is sufficient for most applications. A higher value increases the time an application module instance can exist before it must be removed at the next cleanup. |

**Table 9-10 (Cont.) AM Pool Tuning Parameters**

| Parameter | Description |
|---|---|
| Idle Instance Timeout<br>`jbo.ampool.maxinactiveage` | 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.<br>The default is `600000` milliseconds of idle time (which is 600 seconds or ten minutes). A lower value results in more application module instances being marked as a candidate for removal at the next resource cleanup. A higher value results in fewer application module instances being marked as a candidate for removal at the next resource cleanup. |
| Pool Polling Interval<br>`jbo.ampool.monitorsleepinterval` | The length of time in milliseconds between pool resource cleanup.<br>While the number of application module instances in the pool never exceeds the maximum pool size, available instances which are candidates for getting removed from the pool do not get **cleaned up**until the next time the application module pool monitor wakes up to do its job.<br>The default is to have the application module pool monitor wake up every `600000` milliseconds (which is 600 seconds or ten minutes). Configuring a lower interval results in inactive application module instances being removed more frequently to save memory. Configuring a higher interval results in less frequent resource cleanups. |

**Table 9-10 (Cont.) AM Pool Tuning Parameters**

| Parameter | Description |
|---|---|
| Failover<br>`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:**<br>When you enable application module state passivation, a failure can occur when Oracle WebLogic Server is configured to forcibly release the 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.<br><br>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.<br><br>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. |
| Locking Mode<br>`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. |
| Database Connection Pooling<br>`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. |

**Table 9-10 (Cont.) AM Pool Tuning Parameters**

| Parameter | Description |
|---|---|
| Transaction Disconnect Level<br>`jbo.txn.disconnect_level` | When used in conjunction with `jbo.doconnectionpooling=true`, it specifies BC4J behavior for maintaining JDBC ResultSets. By default, `jbo.txn.disconnect_level` is `0`and you can use passivation 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. |

For parameters that can be configured for memory-constrained systems, see Table 9-11.

**Table 9-11 AM Pool Sizing Configurations - Memory Considerations**

| Parameter | Description |
|---|---|
| Initial Pool Size<br>`jbo.ampool.initpoolsize` | Set this to a low value to conserve memory at the cost of slower performance when additional AM instances are required. The default value of 0 (zero) does not create any AM instances when the AM pool is initialized. |
| Maximum Pool Size<br>`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 while accessing the application if no AM instances are available. |
| Minimum Available Pool Size<br>`jbo.ampool.minavailablesize` | Set to 0 (zero) to shrink the pool to contain no instances when all instances have been idle for longer than the idle time out after a resource cleanup. However, a setting of 1 is commonly used to avoid the costs of re-creating the AM pool. |
| Maximum Available Pool Size<br>`jbo.ampool.maxavailablesize` | Configure this to leave the maximum number of available instances specified after a resource cleanup. |

For parameters that can be configured to reduce the load on the CPU to some extent through a few parameters, see Table 9-12.

**Table 9-12 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. |

**Table 9-12 (Cont.) AM Pool Sizing Configurations - CPU Considerations**

| Parameter | Description |
|---|---|
| `jbo.recyclethreshold` | Configure this value to maintain the AM instance's affinity to a user's session. Maintaining this affinity as much as possible saves the CPU processing cost of needing to switch an AM instance from one user session to another. |

### 9.3.1.5.2 Configuring Application Module Pool Sizing

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

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

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

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

Replace these lines with the following:

```
JAVA_OPTIONS="-Djbo.ampool.doampooling=true
-Djbo.ampool.minavailablesize=1
-Djbo.ampool.maxavailablesize=120
-Djbo.recyclethreshold=60
-Djbo.ampool.timetolive=-1
-Djbo.load.components.lazily=true
-Djbo.doconnectionpooling=true
-Djbo.txn.disconnect_level=1
-Djbo.connectfailover=false
-Djbo.max.cursors=5
-Doracle.jdbc.implicitStatementCacheSize=5
-Doracle.jdbc.maxCachedBufferSize=19 ${JAVA_OPTIONS}"
```

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

### 9.3.1.5.3 Configuring Application Module Pool Resource Cleanup

These parameters affect the frequency and characteristics for AM pool resource cleanups.

For memory-constrained systems, configure the AM pool to clean up more AM instances more frequently so that the memory consumed by the AM instance can be freed for other purposes. However, reducing the number of available AM instances and increasing the frequency of cleanups can result in higher CPU usage and longer response times. See Table 9-13 for more information.

**Table 9-13 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 recreating the AM pool. |
| `jbo.ampool.maxavailablesize` | A lower value generally results in more AM instances being removed from the pool during 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 9-14 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. |

#### 9.3.1.5.4 Designing an Application Module

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 the functionality that a user needs.
- Nest smaller application modules under a single root application module so that the same database connection can be shared among the nested application modules.
- Use lazy loading for application modules. In the Application Module tuning section, customize runtime instantiation behavior to use lazy loading. Lazy loading can also be set JVM-wide by adding the following JVM argument:

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

### 9.3.1.6 Using ADFc Regions

Adding regions to a page can be a powerful addition to the application. While there is no limit to the number of remote regions that you can render in a JSF page, use this capability with caution. For simple pages, where tabs are not used, regions may be combined in the page such that the maximum number of regions is determined by the design of the region and the view object queries it executes. Alternatively, for complex pages that use tabs, limit the use of regions to achieve best performance. For complex tabbed pages, ADF does not deactivate task flow transactions once a region is loaded. When switching tabs, the ongoing transaction must be stopped to achieve best performance.

### 9.3.1.7 Deferring Task Flow Execution

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

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

### 9.3.1.8 Deferring Task Flow Creation in Popups

By default, the child components under a pop-up are created even when pop-up 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 task flow

  > **Note:**
  >
  > This approach cannot be used if any of the following tags are present inside the pop-up:
  >
  > – `f:attribute`
  >
  > – `af:setPropertyListener`
  >
  > – `af:clientListener`
  >
  > – `af:serverListener`
  >
  > It also cannot be used if you need to refer to any child components of the popup before the popup is displayed. Setting `childCreation="deferred"` postpones 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.

- Use conditional activation

  Add property listener on the popup in the **jsff** to set a condition

  Set `activation="conditional"` on the task flow

  Set `activate=`*`condition`* on the task flow

### 9.3.1.9 Configuring the Task Flow Inside Switcher

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

### 9.3.1.10 Reusing Static Data

If the application contains static data that can be reused across the application, the cache data can be collected by using a shared application module. For more information on creating and using shared application modules, see Sharing Application Module View Instances in *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework*.

### 9.3.1.11 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. For more information on specifying preconditions for validation, see How to Set Preconditions for Validation in *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework*.

# 10
# Tuning Oracle TopLink

You can tune EclipseLink, an open-source persistence framework used with Oracle TopLink, to optimize its performance as the Java Persistence API (JPA) implementation.

- 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.
- Basic Tuning Considerations
  To achieve optimal performance, you can follow the tuning recommendations that apply to your own use case scenarios.
- Advanced Tuning Considerations
  After you have performed the recommended modifications, you can make additional changes that are specific to your deployment. Consider carefully whether the advanced tuning recommendations are appropriate for your environment.

## 10.1 About Oracle TopLink and EclipseLink

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

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

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

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

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

# 10.2 Basic Tuning Considerations

To achieve optimal performance, you can follow the tuning recommendations that apply to your own use case scenarios.

- SQL Statement and Query Tuning Parameters
- Cache Configuration Tuning Parameters
- About Mapping and Descriptor Configurations
- About Data Partitioning

## 10.2.1 SQL Statement and Query Tuning Parameters

Table 10-1 and Table 10-2 show tuning parameters and performance recommendations related to SQL statements and querying.

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

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `Parameterized SQL Binding` | By 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. Use the persistence property `eclipselink.jdbc.bind-parameters` in the `persistence.xml` file to configure this.<br>See also "Understanding Caching" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDEFHHEH` and "Understanding Querying" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/queries.htm#CHDGGCJB`<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. |

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

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `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 by using a database.<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>Set this option in an Oracle Weblogic data source by setting `Statement Cached Type` and `Statement Cached Size` configuration options.<br>See also Increasing Performance with the Statement Cache in *Oracle Fusion Middleware Administering JDBC Data Sources for Oracle WebLogic Server*.<br>Default Value: The Oracle Weblogic Server data source default statement cache size is 10 statements per connection. | If your JDBC driver supports this option, you should always enable statement caching. The Oracle JDBC driver supports this option. |
| `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. |

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

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `Batch Writing` | Batch writing can improve the database performance by sending groups of `INSERT`, `UPDATE`, and `DELETE` statements to the database in a single transaction, rather than individually.<br>Use the persistence property `"eclipselink.jdbc.batch-writing"="JDBC"` in the `persistence.xml` file 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: If using weaving (Java EE default) `AttributeLevel`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. |
| `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 by using the `@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 preallocation enables a batch of ids to be queried from the database simultaneously to avoid accessing the database for an id on every insert.<br>Default Value: `50`. | Always use sequence number preallocation for best performance for inserts. `SEQUENCE` or `TABLE` sequencing should be used for optimal performance, not `IDENTITY`, which does not allow preallocation. |

- Entity Relationships Query Tuning Parameters

### 10.2.1.1 Entity Relationships Query Tuning Parameters

Table 10-2 shows the entity relationship between the query parameters for performance tuning.

**Table 10-2 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>Batch fetching has three types: `JOIN`, `EXISTS` , and `IN`. The type is set through the query hint `eclipselink.batch.type`.<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 by using the `@BatchFetch` annotation.<br>Default Value: `Off`. | Use it to query the tables with columns mapping to the table data you need. You should only use either batch fetching or joining if you know that you are going to access all the data; if you do not intend to access the relationships, then let the indirection defer their loading.<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. |

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

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `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>Use this feature to improve query performance by reducing database access. By default, relationships are not join-read; if you are using lazy-loading, each relationship is fetched separately when accessed or as a separate database query if you are not using lazy-loading.<br>You can specify the use of join in `JPQL (JOIN FETCH)`, or you can set it as multilevel in the query hint `eclipselink.join-fetch`. It also can be set in the mapping annotation `@JoinFetch`.<br>Joining is part of the JPA specification, whereas batch fetching is not. And, joining works on queries that do 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 or max results, whereas batch fetching does not.<br>See also "Join Fetching" at `http://www.eclipse.org/eclipselink/documentation/2.5/solutions/performance001.htm#CHDEGCHH`<br>Default Value: Not Used. | Use it to query the tables with columns mapping to the table data you need. You should only use either batch fetching or joining if you know that you are going to access all the data; if you do not intend to access the relationships, then let the indirection defer their loading. For the best performance of selects, where batch fetching is not supported, a join is recommended |

**Table 10-2 (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 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>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>See also "Using Lazy Loading" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/mappingintro001.htm#CEGBCJAG`.<br>Default Value: `On` for collection mapping (`ToMany` mappings, `@OneToMany` and `@ManyToMany`)<br>Default Value: `Off` for reference (`ToOne` mappings, `@OneToOne` and `@ManyToOne`)<br>**Note:** Setting lazy loading to `On` for `@OneToOne` and `@ManyToOne` requires weaving, which is set to `On` by default for Java EE. | Use lazy loading for all mappings. Using lazy loading and querying the referenced objects by using batch fetching or Join is more efficient than Eager loading.<br>You may also consider using optimized loading with `LoadGroups`, which allows a query to force instantiation of relationships. |

## 10.2.2 Cache Configuration Tuning Parameters

You can tune the default internal cache that is provided by EclipseLink. Oracle Toplink or EclipseLink can also be integrated with Oracle Coherence. For information on configuring and tuning an EclipseLink Entity Cache by using Oracle Coherence, see .

The default settings for EJB3.0/JPA that is used with the EclipseLink persistence manager and cache are no locking, no cache refresh, and cache-usage `DoNotCheckCache`. To ensure that your application uses the cache and does not read stale data from the cache (when you do not have exclusive access), you must configure these and other isolation related settings appropriately. Table 10-3 shows the cache configuration options.

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

> **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, or applications that are modifying the data. If your application does not have exclusive access to the data, then you must change some of the defaults from Table 10-3.

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

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `Object Cache` | 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 that are 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 in the `persistence.xml` file 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. |

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

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `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 "Understanding Caching" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDEFHHEH` and "JPA Query Customization Extensions" at `http://www.eclipse.org/eclipselink/documentation/2.5/jpa/extensions/queryhints.htm#sthref498`.<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 10-3 (Cont.) EJB3.0 JPA Entities and Cache Configuration Options**

| Tuning Parameter | Description | Performance Notes |
|---|---|---|
| `Cache Size` | Cache size can be configured through the following persistence properties: `eclipselink.cache.size.`*`entity`*<br>`eclipselink.cache.size.default`<br>`eclipselink.cache.type.default`<br>See also "About the Persistence Unit" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/appdeployment002.htm#BABHCJDG` and "Class PersistenceUnitProperties" at `http://www.eclipse.org/eclipselink/api/2.3/org/eclipse/persistence/config/PersistenceUnitProperties.html`.<br>Default Value: Type `SoftWeak`, Size `100` (per Entity). The default value may be different if Toplink is running on Exalogic. See Enable the Exalogic Automated Tuner in the *Oracle Fusion Middleware Solutions Guide for Oracle TopLink* for more information about the Exalogic default. | Based on your tolerance for stale data, 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.<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 10-4: `No Locking`, `Optimistic`, `Pessimistic`, and `Read Only`.<br>Locking is set through JPA `@Version` annotation, `eclipselink.read-only`<br>See "Descriptors and Locking" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/descriptors002.htm#CHEEEIEA`.<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 10-3 (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>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>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 by using `CheckCacheByExactPrimaryKey`.<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, which are 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>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>• Locking mode<br>• Use of the Session Cache<br>• External Applications<br>• Database Login method `setTransactionIsolation`<br>See also Isolated Cache at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache001.htm#CDEEGICF`. | |

**Table 10-3 (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>See About Cache Refreshing .<br>See also Understanding Caching at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/cache.htm#CDEFHHEH.`<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 |

- About Cache Refreshing
- Locking Mode Policy Options

## 10.2.2.1 About Cache Refreshing

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 refreshes stale objects in the cache on locking errors.

### 10.2.2.2 Locking Mode Policy Options

The locking modes, as shown in Table 10-4, along with EclipseLink cache-usage and query refreshing options, ensures data consistency for EJB entities using JPA. The different combinations have both functional and performance implications, but often the functional requirements for up-to-date data and data consistency lead to the settings for these options, even when it may be at the expense of performance.

For more information, see "Descriptors and Locking" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/descriptors002.htm#CHEEEIEA`.

**Table 10-4 Locking Mode Policies**

| Locking Option | Description | Performance Notes |
|---|---|---|
| `No Locking` | The application does not prevent users overwriting each other's changes. This is the default locking mode. Use this mode if the entity is never updated concurrently or concurrent reads and updates to the same rows with read-committed semantics is sufficient.<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 that the data has not changed since the user read the data.<br>See also "Using Optimistic Locking" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/mappingintro005.htm#CEGDIIIB`. | 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. |

**Table 10-4 (Cont.) Locking Mode Policies**

| Locking Option | 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 the unit of work performance.<br>Set at the entity level by using `@ReadOnly` class annotation. Can also be set at the query level through the query hint `eclipselink.read-only`. | Defining an entity as `Read Only` can perform better than an entity that is not defined as `Read Only`, yet does no inserts, updates, or deletes, since it enables EclipseLink to optimize the unit of work performance. Always use `Read Only` for all read-only operations. |

## 10.2.3 About Mapping and Descriptor Configurations

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

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

For information on Mapping, see “Mapping and Descriptors” at `http://www.eclipse.org/eclipselink/documentation/2.5/solutions/performance002.htm#sthref153`.

## 10.2.4 About Data Partitioning

EclipseLink allows you to configure data partitioning by using the `@Partitioned` annotation. Partitioning enables an application to scale information across multiple databases; including clustered databases.

For more information on using `@Partitioned` and other partitioning policy annotations, see "Partitioning Annotations" at `http://www.eclipse.org/eclipselink/documentation/2.5/jpa/extensions/annotations_ref.htm#CACHIHIB`.

# 10.3 Advanced Tuning Considerations

After you have performed the recommended modifications, you can make additional changes that are specific to your deployment. Consider carefully whether the advanced tuning recommendations are appropriate for your environment.

- Integrating with Oracle Coherence
- Analyzing EclipseLink JPA Entity Performance

## 10.3.1 Integrating with Oracle Coherence

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

For example:

- Replace the default EclipseLink L2 cache with Coherence. This provides support for very large L2 caches that span cluster nodes. EclipseLink's default L2 cache improves performance for multithreaded and Java EE server hosted applications that are 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 using EclipseLink JPA with a Coherence Cache, see Grid Cache Configuration in *Oracle Fusion Middleware Integrating Oracle Coherence*.

For details on Oracle Toplink integration with Oracle Coherence, see Integrating Toplink Grid with Oracle Coherence in *Oracle Fusion Middleware Integrating Oracle Coherence*.

## 10.3.2 Analyzing EclipseLink JPA Entity Performance

The following features in EclipseLink can help you analyze your JPA application performance:

- For form monitoring performance, see "Performance Monitoring" at `http://www.eclipse.org/eclipselink/documentation/2.5/concepts/monitoring003.htm#BABJABIH`. Note that this tool is intended to profile and monitor information in a multithreaded server environment.
- For profiling performance, see "Task 1: Measure EclipseLink Performance with the EclipseLink Profiler" at `http://www.eclipse.org/eclipselink/documentation/2.5/solutions/performance002.htm#CHDIAFJI`. Note that this tool is intended for use with single-threaded finite use cases.
- For debugging performance issues and testing, you can view the SQL generated from EclipseLink. To view the SQL, increase the logging level to `FINE` by using the EclipseLink JPA extensions for logging.

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

# Part IV
# SOA Suite Components

The Oracle SOA Suite components need to be tuned for optimal performance.

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

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

- *Oracle Fusion Middleware User's Guide for Oracle B2B*
- *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework*
- Oracle JCA Adapter Tuning Guide in *Oracle Fusion Middleware Understanding Technology Adapters*

The SOA Suite components are documented in the following topics:

- Tuning the SOA Infrastructure
  You can tune the SOA Infrastructure to optimize its performance in managing composites and their lifecycle, service engines, and binding components in Oracle WebLogic Server, by using Work Managers and other tuning parameters.
- Tuning Oracle BPEL Process Manager
  You can tune Oracle Business Process Execution Language (BPEL) Process Manager properties to optimize its performance at the composite, fabric, application, and server levels.
- Tuning Oracle Mediator
  You can tune Oracle Mediator to optimize its performance as the framework for mediation between various providers and consumers of services and events.
- Tuning Oracle Managed File Transfer
  You can tune Managed File Transfer (MFT) to optimize its performance as the managed file gateway.
- Tuning Oracle Business Rules
  You can tune Oracle Business Rules to optimize its performance in enabling automation of business rules and extraction of business rules from procedural logic, such as Java code or BPEL processes.
- Tuning Oracle Business Process Management
  You can tune Oracle Business Process Management to optimize its performance in providing a seamless integration of all stages of the application development life cycle from design-time and implementation to runtime and application management.
- Tuning Oracle Human Workflow
  You can tune Oracle Human Workflow to optimize its performance in handling various aspects of human interaction with a business process.
- Tuning Oracle Business Activity Monitoring
  You can tune Oracle Business Activity Monitoring (BAM) to optimize its performance in monitoring business services and processes in the enterprise.

- Tuning Oracle Service Bus
  You can tune Oracle Service Bus (OSB) to optimize its performance in providing connectivity, routing, mediation, management, and also some process orchestration capabilities between two or more applications.
- Tuning Oracle Enterprise Scheduler Service
  You can tune Oracle Enterprise Scheduler Service (ESS) to optimize its performance in enabling scheduling and running jobs.
- Tuning Oracle Business Intelligence Performance
  You can tune Oracle Business Intelligence to optimize its performance in collecting, presenting, and delivering data.

# 11
# Tuning the SOA Infrastructure

You can tune the SOA Infrastructure to optimize its performance in managing composites and their lifecycle, service engines, and binding components in Oracle WebLogic Server, by using Work Managers and other tuning parameters.



## 11.1 About the SOA Infrastructure

The SOA Infrastructure is a Java EE-compliant application running on Oracle WebLogic Server.

The application manages composites and their lifecycle, service engines, and binding components. For more information, see Introduction to the SOA Infrastructure Application in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

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

## 11.2 Tuning SOA Work Managers

You can perform a few simple checks and configurations to take advantage of Work Managers.

Beginning with Oracle SOA Suite 12*c* (12.2.1), Work Managers handle most SOA-related work threads. For more details on how Work Managers manage threads and

self-tune, see Understanding Work Managers in *Oracle Fusion Middleware Administering Server Environments for Oracle WebLogic Server*.

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

- Volume of incoming requests that you need processed.
- Internal processing requirements, including any SLA expectations for transactions.
- An understanding of the processes you have that do not use Work Managers, such as the Event Delivery Network and most adapters.

Based on the information collected above, you can take advantage of the Work Managers' self-tuning feature.

- Configuring Database Connections with the SOADataSource Property
- Configuring Work Managers with the SOAMaxThreadsConfig Attribute

## 11.2.1 Configuring Database Connections with the `SOADataSource` Property

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

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

To tune the `SOADataSource`, do the following:

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

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

The `SOADataSource` setting is leveraged by the `SOAMaxThreadConfig` configuration that is explained in Configuring Work Managers with the SOAMaxThreadsConfig Attribute. The `SOADataSource` attribute defines the total number of connections that are available to all Work Managers, while the `SOAMaxThreadConfig` attribute defines what percentage of those connections are available to certain categories of Work Managers.

## 11.2.2 Configuring Work Managers with the SOAMaxThreadsConfig Attribute

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

The number of threads allotted to handle incoming requests, internal processes, and other SOA processes are defined as percentages of the `SOADataSource` property that is

explained in Configuring Database Connections with the SOADataSource Property. The default percentage values and categories of the SOAMaxThreadsConfig attribute are listed in Table 11-1.

**Table 11-1 Thread distributions for Work Managers determined by SOAMaxThreadsConfig**

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

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

To access the attribute:

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

   When you are ready to make your changes, click **Apply**.

Remember that the values you are adjusting on this screen are percentages, not the discrete number of threads. You should ascertain the total number of threads available to you by checking the value of the SOADataSource property, which is described in Configuring Database Connections with the SOADataSource Property.

In a sample scenario, where the SOADataSource attribute is set to 50 connections and if you kept the default SOAMaxThreadConfig percentages that are listed in Table 11-1, you would have the following thread allocations:

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

# 11.3 Tuning SOA Infrastructure Parameters

Tuning SOA infrastructure parameters is important for optimal performance.

Table 11-2 describes the optimal settings for parameters with the greatest impact on SOA Infrastructure performance.

**Table 11-2 Essential SOA Infrastructure Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `AuditLevel`<br>**Default**: `Production` | • High database CPU<br>• Contentions causing increased processing times in applications | To prevent possible performance degradation, decrease the audit level to "off". Set the default value of `Production` only for audit purposes.<br>This parameter can be set in the Enterprise Manager. You can find the `Audit Level` parameter page on the **SOA Infrastructure Common Properties** page.<br>To find this page:<br>1. Toggle the **SOA** folder in your left-hand **Target Navigation**.<br>2. Right-click on the `soa-infra (soa_server)` you want to tune.<br>3. Select **SOA Administration --> Common Properties**<br>For more information about this parameter, see Configuring Oracle SOA Suite and Oracle BPM Suite Profiles in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*. | Keeping the default audit level will generate audit data to be captured in the database and hence cause database growth. Users should use the audit information for debugging errors. |

**Table 11-2 (Cont.) Essential SOA Infrastructure Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `Audit Purge Policy`<br>**Default**: Everyday Midnight and purges records older than 7 days | • Exponential growth in database size<br>• If configured at peak hours, purging can take resources from other processes | • Ensure that auto purge is enabled.<br>• Perform purges more often.<br>• Set the auto purge to kick off at a time when there is less resource contention from other processes.<br>For information on finding the Auto Purge page in the Oracle Enterprise Manager Fusion MIddleware Control, see Deleting Large Numbers of Instances with Oracle Enterprise Manager Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*. | Disabling this feature makes maintaining on-going database growth more time-consuming. |

# 11.4 Using Advanced Tuning Options

You can configure additional performance tuning settings for SOA for specific scenarios.

These options are presented here in no specific order. Before you change any of these properties, you should have a holistic knowledge of your environment, SOA processes, and non-SOA processes.

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

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

- Using Composite Lazy Loading
- Changing Modularity Profiles
- Tuning Your Database for SOA Processes
- Tuning Event Delivery Network Parameters
- Tuning the WebLogic Server

## 11.4.1 Using Composite Lazy Loading

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

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

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

Composite lazy loading is helpful for:

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

Composite lazy loading is enabled by default and can be configured at the domain level and at the composite levels.

- Configuring Composite Lazy Loading for the Domain Level
- Configuring Composite Lazy Loading at the Component Level

### 11.4.1.1 Configuring Composite Lazy Loading for the Domain Level

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

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

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

### 11.4.1.2 Configuring Composite Lazy Loading at the Component Level

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

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

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

Below is a sample code snippet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by Oracle SOA Modeler version 12.2.1.0.0 at [8/7/13 4:14 PM]. -->
<composite name="ValidatePayment"
           revision="1.0"
           label="2013-08-07_16-14-11_843"
           mode="active"
           state="on"
           lazyLoading="false"
           xmlns="http://xmlns.oracle.com/sca/1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
…….
…….
</composite>
```

## 11.4.2 Changing Modularity Profiles

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

12*c* has ready—to—use profiles that can be changed after you complete installation. By default, new 12*c* customers have `SOA_FOUNDATION` as their install profile. Existing customers upgrading to 12*c* have `SOA_CLASSIC` as their install profile by default.

Table 11-3 shows the modularity profiles in the increasing order of memory footprint size.

**Table 11-3 Modularity Profiles**

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

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

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

See, Configuring SOA Infrastructure Properties in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* to find the **SOA Infrastructure Common Properties** page.

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

## 11.4.3 Tuning Your Database for SOA Processes

If needed, you can adopt advanced strategies for tuning your database for SOA processes. Make sure you have already read and followed the general database tuning suggestions covered in Tuning Database Parameters of this book before you progress.

- Collecting Optimizer Statistics
- Tuning Temporary Tablespaces for SOA
- Minimizing SOA Database Contention
- Purging
- Reclaiming Space

### 11.4.3.1 Collecting Optimizer Statistics

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

- Gathering Statistics Automatically
- Gathering Statistics Manually
- Optimizing the MDS Database Repository With Statistics

#### 11.4.3.1.1 Gathering Statistics Automatically

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

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

#### 11.4.3.1.2 Gathering Statistics Manually

Automatic optimizer statistics collection is sufficient for most database objects, but in a database that is close to going live or for tables that are modified or purged

significantly, manual statistic gathering is needed. See Gathering Optimizer Statistics Manually in *Oracle Database SQL Tuning Guide* for more information.

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

#### 11.4.3.1.3 Optimizing the MDS Database Repository With Statistics

Ensure that automatic statistics collection is enabled. See Controlling Automatic Optimizer Statistics Collection in *Oracle Database SQL Tuning Guide* for more information.

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

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

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

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

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

### 11.4.3.2 Tuning Temporary Tablespaces for SOA

See Tuning Database Files for general guidelines on tuning `TEMP` tablespaces for Oracle Fusion Middleware before you progress to this topic.

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

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

### 11.4.3.3 Minimizing SOA Database Contention

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

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

- `DB CPU`
- `Db file sequential read`, `db file scattered read`
- `log file sync`

- `enq: HW - contention`
- `enq: TX - index contention`
- `buffer busy waits`
- `gc buffer busy acquire`, `gc buffer busy release` (RAC)
- `enq: SQ - contention`
- Tuning the Redo Log Performance (log file sync)
- Migrating BasicFiles to SecureFiles (enq:HW - contention)
- Creating Hash Partitioned Indexes (enq: TX - index contention)

### 11.4.3.3.1 Tuning the Redo Log Performance (log file sync)

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

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

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

See Tuning Database Files for general guidelines on tuning redo logs for Oracle Fusion Middleware before using the strategies here to tune for SOA processes.

**Finding LGWR wait events**

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

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

**Sizing Online Redo Logs to Control the Frequency of Log Switches and Minimize System Waits**

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

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

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

**Optimizing the Redo Log Disk to Prevent Bottlenecks**

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

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

**Determining the Optimal Sizing of the log_buffer**

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

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

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

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

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

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

**Tuning the LGWR Process**

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

**Using Smart Flash Logging for ExaData**

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

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

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

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

### 11.4.3.3.2 Migrating BasicFiles to SecureFiles (enq:HW - contention)

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

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

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

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

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

- Set the database parameter `SECURE_FILES = ALWAYS`.

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

- Use the online redefinition method.

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

  Using the online redefinition method to migrate to SecureFiles can be done with very little downtime.

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

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

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

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

**Table 11-4 LOB Storage Attributes**

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

### 11.4.3.3.3 Creating Hash Partitioned Indexes (enq: TX - index contention)

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

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

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

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

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

## 11.4.3.4 Purging

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

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

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

### 11.4.3.5 Reclaiming Space

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

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

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

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

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

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

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

## 11.4.4 Tuning Event Delivery Network Parameters

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

Table 11-5 lists parameters that you can find in the Fusion Middleware MBean Browser and tune for improved event delivery.

**Table 11-5 Event Delivery Network Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `numberOfPollerThreads`<br>**Default**: -1 | • Out-of-resource issues, for example, out of memory, system overload, transaction issue, and so on.<br>• Contention with other SOA threads | The default value of -1 means that the system uses `ThreadsPerSubscriber` to determine a poller thread count. This is optimal for most configurations.<br>However, if you have a high number of subscribers, the default setting tries to assign a thread to each subscriber. This slows your system down. You should define a positive integer to limit the amount of poller threads created for this task.<br>See Updating the Local numberOfPollerThreads Value at the Service Component Level in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* for information on how to change this parameter's value in the Fusion Middleware MBean Browser. | If the value is too low for your system, then poller threads can cause event backlogs and long latencies between event publishing and composite instance creation.<br>If the value is too high, then excess poller threads consume the server's resources needlessly. |
| `ThreadsPerSubscriber`<br>**Default**: 1 thread | • Out-of-resource issues, for example, out of memory, system overload, transaction issue, and so on.<br>• Contention with other SOA threads | Typically, the default of 1 thread per subscriber is optimal.<br>Note that `numberOfPollerThreads` should be adjusted first, since that parameter takes precedence over this value.<br>See Updating the ThreadsPerSubscriber Attribute in the System MBean Browser in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* for information on how to change this parameter's value in the Fusion Middleware MBean Browser. | If the value is too low for your system, then poller threads can cause event backlogs and long latencies between event publishing and composite instance creation.<br>If the value is too high, then excess poller threads consume the server's resources needlessly. |

Table 11-6 lists the parameters that you can modify for individual business events in JDeveloper. To modify these attributes, right-click the event that you want to edit to bring up the pop-up menu. From this menu, select **Edit Subscribed Events...** or **Edit Published Events...**, depending on the parameter that you are trying to edit.

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

**Table 11-6 Business Event Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| Consistency for a Subscribed Event<br>**Default**: `oneAndOnlyOne` | You are experiencing either one or both problems with business event delivery.<br>• Unfulfilled delivery guarantee requirements to event subscribers<br>• Unnecessary system overhead from global transactions | Set the level for a selected business event to `guaranteed` in JDeveloper. A guaranteed delivery is performed in a local transaction with only one trip to the main queue.<br>You can also edit this parameter on the **Subscriptions** page in the Oracle Enterprise Manager Fusion Middleware Control. See Viewing Business Event Subscribers in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* for details. | The `oneAndOnlyOne` parameter guarantees delivery by taxing resources.<br>If a `guaranteed` delivery fails, then there are no local retries and a system failure message is generated. Message duplication could occur in the event that the calling global transaction rolls back and retries since the message delivery is outside of that transaction. |
| Durability for a Subscribed Event<br>**Default**: `Yes` | You are experiencing either one or both problems with business event messages.<br>• Multiple dropped events<br>• Unnecessary retention of messages in the system | Set the value under the `Durable` column to `No` to disable durability for a subscribed event by using JDeveloper. This frees the system from having to persist messages to storage. | If the subscriber is not running when events are published, setting the value to `No` causes the system to drop events.<br>Setting the value to `Yes` retains events in the JMS server and incurs overhead. |
| Persistent Delivery for a Published Event<br>**Default**: `yes` | • Unreliable messaging<br>• High overhead | Set this value to `No` to disable persistent delivery. This reduces overhead. | Setting the value to `No` causes less reliable messaging following an event publish since there is no persistence.<br>Setting the value to `Yes` incurs overhead by guarding against a JMS server crash. |

**Table 11-6 (Cont.) Business Event Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| Time-to-live for a Published Event<br>**Default**: 0 ms | • Non-expired and unconsumed messages are occupying system resources and requiring manual cleanup. | Specify a positive integer so that expired messages are automatically removed from the system and not consumed by the subscribers. The integer represents milliseconds. | If the message expiration duration value is too low, published messages can expire before an intended subscriber can read it. Once it is gone, it cannot be retrieved. |
| | • Messages are deleted before subscribers can read them. | The best value depends on your system and can be determined by monitoring metrics. | If the value is too high, then lingering messages can occupy system resources. |

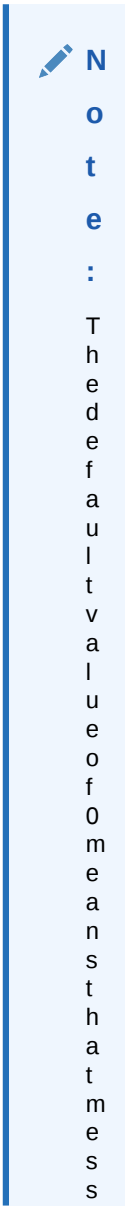
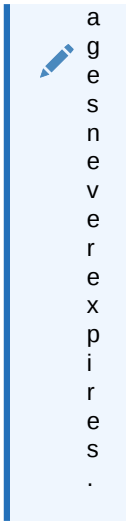


**Table 11-6 (Cont.) Business Event Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| | |  | |



- Adding JMS Topics with Mapping

## 11.4.4.1 Adding JMS Topics with Mapping

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

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

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

- Choosing a JMS Topic Type
- Creating JMS Topics
- Mapping Events to JMS Topics

### 11.4.4.1.1 Choosing a JMS Topic Type

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

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

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

#### 11.4.4.1.2 Creating JMS Topics

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

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

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

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

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

#### 11.4.4.1.3 Mapping Events to JMS Topics

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

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

## 11.4.5 Tuning the WebLogic Server

The performance of the SOA Infrastructure depends on the WebLogic Server. Though tuning the WebLogic Server is a separate task not thoroughly addressed in this book, you can use Table 11-7 to check the tuning knobs that affect the SOA Infrastructure.

**Table 11-7 Essential WebLogic Server Tuning for SOA Infrastructure**

| Parameter | Tuning Recommendation | Resource |
|---|---|---|
| `ProductionModeEnabled`<br>**Default**: The mode you set during domain creation. | Production mode maximizes performance. You should enable this if you are not developing applications. You can enable the `ProductionModeEnabled` MBean in Oracle Fusion Middleware Control. | See Configure General Settings in *Oracle Fusion Middleware Administering Oracle WebLogic Server with Fusion Middleware Control*.<br>Changing the domain mode also changes certain security and autodeployment settings. See Development vs. Production Mode Default Tuning Values in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server* for more information on domain modes. |
| WebLogic Server Logging Levels<br>**Default**: `Notification` | To reduce the volume of logging requests, use the lowest acceptable logging level, such as `ERROR` or `WARNING` whenever possible. You can set log levels for handlers and loggers in a variety of ways. | See Using Log Severity Levels in *Oracle Fusion Middleware Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server* for these methods. |
| HTTP Access Logging<br>**Default**: Enabled | By default, the HTTP subsystem keeps a log of all HTTP transactions in a text file. Turn off HTTP access logging to improve performance. You can disable this property by using the Oracle WebLogic Server Administration Console. | See "Enable and configure HTTP logs" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*. |
| JMS Persistence and Persistence Storage<br>**Default**: `Enabled` | Ensure that the right persistence level is set for the Java Message Service (JMS) destinations.<br>• 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.<br>• For non-persistent JMS scenarios, 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 in the WebLogic Server console. You can also override the persistence mode at the JMS destination level. | See Using Custom File Stores and JDBC Stores in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server* for more information on creating and managing persistent JMS stores. |

**Table 11-7 (Cont.) Essential WebLogic Server Tuning for SOA Infrastructure**

| Parameter | Tuning Recommendation | Resource |
|---|---|---|
| Connection Backlog Buffering | You can tune the `Accept Backlog` parameter when dealing with a large number of concurrent clients.<br>The `Accept Backlog` parameter specifies how many TCP connections can be buffered in a wait queue. You can tune the number of connection requests that a WebLogic Server instance accepts before refusing additional requests. | For more information, see Tuning Connection Backlog Buffering in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*. |

# 11.5 Advanced Tuning for Work Managers

Work Managers are mapped to SOA projects and specific components, and you can use some advanced configuration options to fine-tune the Work Manager performance.

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

High priority composites can be associated with a Work Manager group that has been configured for higher priority. Table 11-8 lists the set of Work Managers that are created when SOA is installed and describes the work area they manage.

**Table 11-8 Work Manager Descriptions**

| Work Manager Name | Responsible Area |
|---|---|
| `SOA_Request_WM` | SOA synchronous request clients, such as the following:<br>• Facade invocation<br>• WebService client requests<br>• Direct/ADF/Rest requests<br>• B2B |
| `SOA_Notification_WM` | All SOA notification requests. |
| *`WorkManagerName`*`_dspSystem` | BPEL-specific system dispatcher messages. |
| *`WorkManagerName`*`_dspInvoke` | BPEL-specific engine process invocation dispatcher messages. |
| *`WorkManagerName`*`_dspEngine` | BPEL engine process dispatcher messages. |
| *`WorkManagerName`*_dspNonBlocking | BPEL engine process non-blocking invocation dispatcher messages. |
| *`WorkManagerName`*__Analytics | BPEL analytics. |
| *`WorkManagerName`*`_MediatorParallelRouting` | Mediator parallel routing. |
| *`WorkManagerName`*`_MediatorErrorHandling` | Mediator error handling. |
| *`WorkManagerName`*`_bpmnSystem` | BPM system dispatcher messages. |
| *`WorkManagerName`*`__bpmnInvoke` | BPM engine process invocation dispatcher messages. |

**Table 11-8 (Cont.) Work Manager Descriptions**

| Work Manager Name | Responsible Area |
|---|---|
| *WorkManagerName*__bpmnEngine | BPM process engine dispatcher messages. |
| *WorkManagerName*__bpmnNonBlocking | BPM process non-blocking invocation dispatcher messages. |
| SOA_DataSourceBound_WM | All SOA backend processing services that access.SOADataSource, including Workflow Enterprise JavaBeans (EJBs). |
| SOA_Default_WM | All SOA services that do not access the SOADataSource connection pool. It also handles Case Management. |
| SOA_EDN_WM | Event Delivery Network (EDN). |
| *WorkManagerName*_Adapter | Adapter framework. |

The SOAMaxThreadsConfig property, discussed in Configuring Work Managers with the SOAMaxThreadsConfig Attribute, determines the number of connections that are used by Work Managers to process incoming requests, internal processes, and other processes. This configuration determines the optimal usage for each of these processing categories when the system is functioning at its full potential.

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

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

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

- Configuring Fair Share Request Class for SOA Work Managers
- Creating a New Work Manager Constraint

## 11.5.1 Configuring Fair Share Request Class for SOA Work Managers

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

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

## 11.5.2 Creating a New Work Manager Constraint

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

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

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

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

# 12
# Tuning Oracle BPEL Process Manager

You can tune Oracle Business Process Execution Language (BPEL) Process Manager properties to optimize its performance at the composite, fabric, application, and server levels.

- About BPEL Process Manager
  Oracle BPEL Process Manager offers a comprehensive and easy-to-use infrastructure for creating, deploying, and managing BPEL business processes.
- Tuning BPEL Parameters
  You can tune BPEL parameters for optimal performance.
- Using Other Tuning Strategies
  You can locate the Oracle BPEL Process Manager tables that are impacted by instance data growth and purge them for optimal performance.

## 12.1 About BPEL Process Manager

Oracle BPEL Process Manager offers a comprehensive and easy-to-use infrastructure for creating, deploying, and managing BPEL business processes.

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.

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

## 12.2 Tuning BPEL Parameters

You can tune BPEL parameters for optimal performance.

Tuning recommendations for BPEL parameters described here are *likely* or *highly likely* to improve performance. For descriptions of the other tuning parameters available for SOA Components, see the component-specific topics in this guide.

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

- Tuning BPEL Engine
- Tuning BPEL in a Composite

## 12.2.1 Tuning BPEL Engine

You can configure the performance tuning properties at the BPEL engine level by using the Enterprise Manager Fusion Middleware Control. For information on using Oracle Enterprise Manager Fusion Middleware Control to configure and monitor parameters, see Getting Started with Administering Oracle SOA Suite and Oracle BPM Suite and Accessing the System MBean Browser from the Component Property Pages in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

- Tuning BPEL Engine Parameters

### 12.2.1.1 Tuning BPEL Engine Parameters

Table 12-1 lists the essential tuning parameter that you can adjust to improve performance for the BPEL engine.

**Table 12-1 Essential BPEL Engine Tuning**

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

Table 12-2 describes additional BPEL engine parameters that can be tuned for small performance improvements. Note that for most use cases, the default value is the recommended value.

**Table 12-2 Other BPEL Engine Tuning Knobs**

| Parameter | Description |
|---|---|
| `SyncMaxWaitTime`<br>**Default**: `45` seconds. | You can decrease this parameter's value to improve performance.<br>The `SyncMaxWaitTime` property sets the maximum time the process result receiver waits for a result before returning. This property is required for synchronous interactions and is applicable to transient processes.<br>See How To Specify Transaction Timeout Values in *Oracle Fusion Middleware Developing SOA Applications with Oracle SOA Suite* for instructions on how to find this property in the System MBean Browser of Oracle Enterprise Manager Fusion Middleware Control. |

**Table 12-2 (Cont.) Other BPEL Engine Tuning Knobs**

| Parameter | Description |
|---|---|
| `largedocumentthreshold`<br>**Default**: `10000` (100 kilobytes). | You can decrease this parameter's value to improve performance.<br>This property sets the maximum size (in kilobytes) of a BPEL variable before it is stored in a separate table from the rest of the instance scope data. It is applicable to both durable and transient processes.<br>Large XML documents can slow down the performance if they are constantly used while processing an instance.<br>See Configuring BPEL Process Service Engine Properties in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* to see how to find and tune this parameter in the Enterprise Manager Fusion Middleware Control. |
| `validateXML`<br>**Default**: `False`. | You should set this parameter to the default value of `False` to improve performance.<br>This property can make the Oracle BPEL Process Manager intercept nonschema-compliant payload data by validating incoming and outgoing XML documents. However, XML payload validation can slow performance.<br>You can find this parameter in the System MBean Browser. See Configuring BPEL Process Service Engine Properties in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* for information on how to find advanced BPEL properties by using the **More BPEL Configuration Properties...** button from the **BPEL Service Engine Properties** page in Enterprise Manager Fusion Middleware Control. |
| `InstanceKeyBlockSize`<br>**Default**: `10000` keys. | You can increase the instance key block size to a value greater than the number of updates to the `ci_id_range` table to improve performance.<br>The `InstanceKeyBlockSize` property controls the instance ID range size. Oracle BPEL Server creates instance keys (a range of process instance IDs) in batches by 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.<br>See Configuring BPEL Process Service Engine Properties in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* to see how to find and tune this parameter by using the System MBean Browser in Enterprise Manager Fusion Middleware Control. |

**Table 12-2 (Cont.) Other BPEL Engine Tuning Knobs**

| Parameter | Description |
|---|---|
| Audit Level Threshold<br>**Default**: `10000`. | You can decrease this parameter's value to improve performance.<br>This property sets the maximum size (in kilobytes) of an audit trail details string before it is stored separately from the audit trail. Strings larger than the threshold setting are stored in the `audit_details` table instead of the `audit_trail table`. In cases where the variable is very large, performance can be severely impacted by logging it to the audit trail.<br>See Configuring BPEL Process Service Engine Properties in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite* to see how to find and tune this parameter in Enterprise Manager Fusion Middleware Control. |

## 12.2.2 Tuning BPEL in a Composite

You can tune BPEL properties for individual composites to improve performance. The BPEL properties set inside a composite affect the behavior of the component containing the BPEL process only. Each BPEL process can be created as a component of a composite.

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

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

**Table 12-3 Essential BPEL in a Composite Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `OneWayDeliveryPolicy`<br>**Default**: `async.persist` | Slow performance because resources are being used to persist delivery messages. | Set value to `async.cache`. Incoming delivery messages for durable processes are kept only in the in-memory cache.<br>By default, incoming requests are saved in the delivery service database table `dlv_message`. | This setting has a high risk of losing messages or overloading the system. It also changes the threading model for adapter. |

**Table 12-3 (Cont.) Essential BPEL in a Composite Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `Audit Policy`<br>**Default**: `All activities` | Slow performance because every activity is being audited. | Audit only key activities. | Lower level activities do not have an audit trail. |
| `inMemoryOptimization`<br>**Default**: `False` | Slow performance because the `completionPersistPolicy` parameter has been activated at the BPEL component level, causing the BPEL server to dehydrate either all or some instances. | Set value to `False` to tell the Oracle BPEL Server that this process is a transient process and dehydration is not required. | No dehydration means that activities in the instance are lost if the system crashes. |

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

**Table 12-4 Other BPEL in a Composite Tuning Knobs**

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

# 12.3 Using Other Tuning Strategies

You can locate the Oracle BPEL Process Manager tables that are impacted by instance data growth and purge them for optimal performance.

- Identifying Tables Impacted By Instance Data Growth

## 12.3.1 Identifying Tables Impacted By Instance Data Growth

Instance data occupies space in Oracle BPEL Process Manager schema tables. Data growth from auditing and dehydration can have a significant impact on database performance and throughput.

You can use Table 12-5 to locate tables that may be affected by instance data growth. See Monitoring Space Usage, Hardware Resources, and Database Performance in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business*

*Process Management Suite* for advice on how to monitor performance for the following database tables:

**Table 12-5 Oracle BPEL Process Manager Tables Impacted by Instance Data Growth**

| Table Name | Table Description |
|---|---|
| audit_trail | Stores the audit trail for instances. The audit trail viewed in Oracle BPEL Control is created from an XML document. As an instance is processed, each activity writes events to the audit trail as XML. |
| audit_details | Stores audit details that can be logged through the API. Activities such as an assign activity logs 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. |
| 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 the 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. |

When you have determined which tables are causing slow performance, you can purge them. See Understanding Growth Management Challenges and Testing Strategies in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle*

*Business Process Management Suite* for more information on managing database growth.

# 13
# Tuning Oracle Mediator

You can tune Oracle Mediator to optimize its performance as the framework for mediation between various providers and consumers of services and events.

- About Oracle Mediator
  Mediator is a component of the Oracle SOA Suite offering that provides mediation capabilities like selective routing, transformation and validation capabilities, along with various message exchange patterns, like synchronous, asynchronous and event publishing or subscription.
- Tuning Mediator Parameters
  You can tune the Oracle Mediator properties to improve performance if necessary.
- Using Resequencer for Messages
  A Resequencer is used to rearrange a stream of related but out-of-sequence messages back into order.

## 13.1 About Oracle Mediator

Mediator is a component of the Oracle SOA Suite offering that provides mediation capabilities like selective routing, transformation and validation capabilities, along with various message exchange patterns, like synchronous, asynchronous and event publishing or subscription.

Oracle Mediator provides the framework to mediate between various providers and consumers of services and events. The Mediator service engine runs with the SOA Service Infrastructure Java EE application.

> **Note:**
>
> For details about the SOA Suite, see *Oracle Fusion Middleware Developing SOA Applications with Oracle SOA Suite*.
>
> For details about Oracle Mediator, see Administering Oracle Mediator Service Components and Engines in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

## 13.2 Tuning Mediator Parameters

You can tune the Oracle Mediator properties to improve performance if necessary.

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

Table 13-1 describes the parameter values that can be tuned for performance. Note that the need to tune Mediator to improve performance is unlikely.

**Table 13-1 Essential Mediator Tuning Knobs**

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

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

# 13.3 Using Resequencer for Messages

A Resequencer is used to rearrange a stream of related but out-of-sequence messages back into order.

It sequences the incoming messages that arrive in a random order and then sends them to the target services in an orderly manner.

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

**Table 13-2 Essential Tuning Knobs for Resequencer in Mediator**

| Parameter | Tuning Recommendation |
|---|---|
| `ResequencerMaxGroupsLocked`<br>**Default:** 4 rows | Increase the default value to lock more Resequencer groups from the database in one iteration. |

**Table 13-2 (Cont.) Essential Tuning Knobs for Resequencer in Mediator**

| Parameter | Tuning Recommendation |
|---|---|
| `ResequencerLockerThreadSleep`<br>**Default:** 10 seconds | If resequencer groups constitute a small percentage of total groups and messages, increase the default value to perform fewer trips to the database to lock resequencer groups. |
| `DeleteMessageAfterComplete`<br>**Default:** `True` | Set the value to `True` to delete message after successful execution. For a high load use case, this results in more database space.<br>Changing the default value to `False` retains the resequenced messages in the resequencer database. This slows down the resequencer database queries, which in turn degrades the performance. |

See Configuring Resequenced Messages in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

14

# Tuning Oracle Managed File Transfer

You can tune Managed File Transfer (MFT) to optimize its performance as the managed file gateway.

- About Managed File Transfer
  Oracle Managed File Transfer (MFT) is a high performance, standards-based, end-to-end managed file gateway.
- Tuning MFT Parameters
  You can tune MFT parameters to optimize performance.

## 14.1 About Managed File Transfer

Oracle Managed File Transfer (MFT) is a high performance, standards-based, end-to-end managed file gateway.

It features design, deployment, and monitoring of file transfers using a lightweight web-based design-time console that includes file encryption, scheduling, and embedded FTP and sFTP servers.

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

## 14.2 Tuning MFT Parameters

You can tune MFT parameters to optimize performance.

Table 14-1 lists and describes parameters that you likely need to tune to improve MFT performance. To diagnose problem areas in MFT, see Monitoring Oracle Managed File Transfer and Administering Oracle Managed File Transfer in *Oracle Fusion Middleware Using Oracle Managed File Transfer*.

**Table 14-1 Essential MFT Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| Processor count<br>**Default**: 2 for each type of processor | JMS messages are accumulating in message processing queues. | Increase the processor count for the queues where messages are accumulating.<br>The optimal value depends on the meta data and incoming payload. You can calculate the optimal processor count by using DMS metrics.<br>To enable DMS metrics, add the MBean property `enablePerformanceMetric`. To disable metrics later, set the value to `False`. | Having more processors requires more system resources for concurrent processing. |
| Maximum Concurrent Request and Max Logins settings for Embedded FTP/SFTP server<br>**Default**: `10` | • Multiple connection requests in waiting status<br>• The message **Too many users logged in, user will be disconnected** occurs in the embedded server log file | Increase the maximum number of concurrent requests and maximum number of logins for embedded FTP/SFTP server.<br>You can increase the count so long as performance continues to scale linearly.<br>If the embedded server service (FTP/SFTP) is not being used, then disable this setting. | Increased count requires more system resources for concurrent processing. |
| LDAP Max Pool<br>**Default**: `10` | Number of concurrent connections to the LDAP consistently reaches max limit. | Increase count.<br>Because LDAP is a shared resource for all deployed applications in WebLogic server, you should monitor LDAP connections and adjust this value accordingly. | Increased count requires more system resources. |
| Max connections to `MFTDataSource`<br>**Default**: `50` | Number of concurrent connection to the data source consistently reaches max limit. | Increase the connection count so long as performance continues to scale linearly.<br>Optimal value can be determined based on the number of processors, listening source threading model, and max concurrent request settings of embedded servers. | Increased count requires more system resources. |
| Generating checksum setting<br>**Default**: `Enabled` | Overall MFT message processing is slow. | Disable this parameter if checksum validation for delivered payloads from MFT is not necessary. | Generating checksum is a time consuming operation. |

**Table 14-1 (Cont.) Essential MFT Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| Regular purge<br>**Default**: `Disabled` | • Disk space is approaching the maximum limit.<br>• Table space used by MFT tables reaches the max table space allotted. | Run purge to free disk or table space. | Historical information or data is discarded. |

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

**Table 14-2 MFT Parameters with Low or Medium Importance**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| Processing function or callout usage recommendation for broadcasting use cases<br>**Default**: `Target Level` | Associated processing function or callout is executed for each target that degrades the performance. | For broadcasting use cases, associate processing functions or callouts at source level instead of target level as much as possible. | None. |
| Sub-folder count MBean setting<br>**Default**: `256` | Degraded disk performance caused by MFT switching among a high number of sub-folders to store files. | Reduce the sub-folder count. | Reducing the number of sub-folders increases the number of files stored in each sub-folder. If the volume of incoming files is high, the number of the files inside a single sub-folder degrades performance. |
| Store Inline payload setting<br>**Default**: `File System` | Slow performance because accessing inline payload for Web Service sources from the disk takes too much time. | Store inline payload in the database rather than the file system. | The table size used by MFT increases as inline payloads are stored in the database. |
| `Always Save Modified Files` setting at the target level<br>**Default**: `False` | If you have changed this setting to `True` for auditing purposes, you will have increased the disk space usage. | The default value of `False` reduces disk space usage. | No audit information is available.<br>Note that a target level resubmit does not work if there was any pre-processing associated with the target. |
| `minFileSizeForProgressMonitor`<br>**Default**: `10 MB` | Frequent updates about byte transfer. | Specify a minimum file size so that the transfer progress screen appears for larger files only. | For files smaller than the minimum specified, the file transfer progress is not displayed. |
| `progressMonitorTimeToCommit` MBean<br>**Default**: `4` seconds | Frequent updates about byte transfer. | Specify a minimum file size so that the transfer progress screen appears for larger files only. | Database updates on bytes transferred for ongoing file transfers are slower. |

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

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `MaxMdsSessionCacheCount`<br>**Default**: `100` | Out-of-memory exceptions caused by MDS cache memory footprint. | Decrease this value. | Decreasing this will decrease the performance of the overall MFT message processing because accessing data from the cache is faster. |

- Tuning Remote FTP / SFTP/ FILE Type Sources
- Minimizing MDS label
- Adjusting the Materialized Views Refresh Interval

## 14.2.1 Tuning Remote FTP / SFTP/ FILE Type Sources

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

Table 14-3 lists the properties.

**Table 14-3 Tuning Remote FTP/SFTP/FILE Type Source**

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

## 14.2.2 Minimizing MDS label

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

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

- Minimize frequent deployments and meta data creations.
- Use bulk deployment for WLST commands.
- Make all changes for metadata and deploy them at once.

## 14.2.3 Adjusting the Materialized Views Refresh Interval

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

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

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

The materialized views used by MFT are:

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

# 15
# Tuning Oracle Business Rules

You can tune Oracle Business Rules to optimize its performance in enabling automation of business rules and extraction of business rules from procedural logic, such as Java code or BPEL processes.



## 15.1 About Oracle Business Rules

Oracle Business Rules provides an easy-to-use authoring environment as well as a very high-performance inference-capable rules engine.

Oracle Business Rules is part of the Oracle Fusion Middleware stack and is a core component of many Oracle products including both middleware and applications.

See *Oracle Fusion Middleware Designing Business Rules with Oracle Business Process Management* and Getting Started with Oracle Business Rules in *Oracle Fusion Middleware Developing SOA Applications with Oracle SOA Suite*.

## 15.2 Tuning Oracle Business Rules

You can tune Oracle Business Rules to optimise performance.

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

**Table 15-1 Essential Business Rules Tuning Strategies**

| Strategy | Description | Recommendation |
|---|---|---|
| Use Java Beans | The rule engine is most efficient when the facts it is reasoning on are Java Beans (or RL classes) and the associated tests involve bean properties. | The beans should expose the get and set methods (if set is allowed) for each bean property.<br>If application data is not directly available in Java Beans, flatten the data to a collection of Java Beans that are asserted as facts (and used in the rules). |

**Table 15-1 (Cont.) Essential Business Rules Tuning Strategies**

| Strategy | Description | Recommendation |
|---|---|---|
| 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.<br>If the fact type has a hierarchical structure, consider using the `assertXPath` method or other means to assert object hierarchy. |
| Avoid side effects in rule conditions | The tests in a rule condition may be evaluated a greater or lesser number of times than would occur in a procedural program. | Methods or functions, which have side effects such as changing a value or state should not be used in a rule condition.<br>If a method or function has side effects, those side effects may be performed an unexpected number of times. |
| Avoid expensive operations in rule conditions | Expensive operations would include any operation that involves I/O (disk or network) or even intensive computations. These operations should be done externally to the rules engine. | Expensive operations should be avoided in rule conditions. In general, consider avoiding I/O or DBMS access from the rules engine directly.<br>For other expensive operations or calculations, consider performing the computations and assert the results as a Java or RL fact. These facts are used in the rule conditions instead of the expensive operations. |
| Consider pattern ordering | Reordering rule patterns can improve the performance of rule evaluation in time, memory use, or both. Finding the optimal order for your system requires some experimentation. | If a fact is not expected to change or does not change frequently during rule evaluation, order the fact clauses by the expected rate of change from least to greatest.<br>If a fact clause (including any tests that involve only that fact) is expected to match fewer facts than other fact clauses in the rule condition, order the fact clauses from most restrictive (matches fewest facts) to least restrictive. |
| Consider the ordering of tests in rule conditions | Proper ordering can reduce the amount of computation required for facts that do not satisfy the rule condition. | The tests in a rule condition should be ordered so that a more restrictive test occurs before a less restrictive test.<br>If the degree of restrictiveness is not known, or estimated to be equal for a collection of tests, then simpler tests should be placed before more expensive tests. |

- Exerting assertXPath Support

## 15.2.1 Exerting assertXPath Support

The `assertXPath` method asserts the whole hierarchy in one call, but also asserts some XLink facts for children facts to link back to parent facts. Though very convenient, it may have a performance impact.

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

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

If XLink facts for children facts 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 the `assertXPath` method, 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());
}
```

16

# Tuning Oracle Business Process Management

You can tune Oracle Business Process Management to optimize its performance in providing a seamless integration of all stages of the application development life cycle from design-time and implementation to runtime and application management.

- About Oracle Business Process Management
  The Oracle Business Process Management Suite provides an integrated environment for developing, administering, and using business applications centered around business processes.
- Tuning Business Process Management Parameters
  You can tune BPM performance parameters in the Enterprise manager, through the SOA Administration in BPMN properties.
- Using Other Tuning Strategies
  You can consider using the following strategies to further improve performance.

## 16.1 About Oracle Business Process Management

The Oracle Business Process Management Suite provides an integrated environment for developing, administering, and using business applications centered around business processes.

Oracle Business Process Management is layered on the Oracle SOA Suite and shares many of the same product components, including Business Rules, Human Workflow, and Oracle Adapter Framework for Integration.

See,*Oracle Fusion Middleware User's Guide for Oracle Business Process Management*.

For more details on tuning Oracle Business Process Management with your other Oracle Fusion Middleware components, see .

## 16.2 Tuning Business Process Management Parameters

You can tune BPM performance parameters in the Enterprise manager, through the SOA Administration in BPMN properties.

To tune the performance of the Oracle Business Process Management engine, you can reduce resource demands to reduce latency.

To reduce resource demands, you can tune the parameters listed in Table 16-1:

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

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `largedocumentthreshold`<br>**Default**: `10000` (100 kilobytes) | Instances are being processed slowly because you are storing large BPMN Data objects. | Decrease the maximum size (in kilobytes) of this parameter to limit the size of BPMN Data Objects. If they surpass this limit, they are stored in a separate location from the rest of the instance scope data.<br>This property is applicable to both durable and transient processes. | The overflow data is stored in an external append-only table. This adds to overall database size and can increase the overall workload when loading instances from the database. |
| `auditLevel`<br>**Default**: `Inherit from Infrastructure` | You are seeing frequent database inserts into the `audit_trail` table. These are caused by audit events being logged by a process. | Reduce or disable audit. You can switch to any of the following settings:<br>• `Off` to log no events or audit events<br>• `Minimal` to log only events<br>• `Error` to log only serious problems<br>You can also consider expanding the size of the `AuditKeyExtents`. | You lose granular error reporting that you could use to diagnose problems later. Always choose the audit level according to your business requirements and use cases.<br>For more information on how to use audit trails for monitoring, see Monitoring BPMN Process Service Components and Engines in *Oracle Fusion Middleware Administering Oracle SOA Suite and Oracle Business Process Management Suite*. |

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

# 16.3 Using Other Tuning Strategies

You can consider using the following strategies to further improve performance.

- Tuning Oracle Workspace Applications
- Tuning Process Measurement

## 16.3.1 Tuning Oracle Workspace Applications

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

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

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

**Table 16-2 Workspace and Worklist Application Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `HTTP Session Timeout`<br>**Default**: 15 minutes | Memory is being allocated for users who may no longer be actively using the system. | To better manage resource usage, decrease the session timeout value, in minutes, to the smallest value that preserves the expected user experience. This allows the system to reclaim any resources that are associated with unused sessions as soon as possible.<br>This parameter is edited in the in the `web.xml` file. The following is a sample snippet of the`web.xml` file:<br>`<session-config>`<br>`<session-timeout>`<br>**5**<br>`</session-timeout>`<br>`</session-config>` | A short timeout value may mean users have to login more often if they let the time expire. They also may potentially lose session data. |
| `ADF Client State Token`<br>**Default**: 15 | The default value may consume too much memory. | Decrease the value to 3 to minimize the memory footprint.<br>Through this setting, you can control the number of pages users can navigate by 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 the`web.xml` file:<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>` | If the user clicks the **Back** button more than 3 times, there is no session data stored for that page.<br>If the value is too small, users get an error when they click the **Back** button. |

**Table 16-2 (Cont.) Workspace and Worklist Application Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `Compress_View_State Token`<br>**Default**: `True` | Slow performance on slower or higher latency networks. | Set this value to **True** to enable zipping. By default, this value is set to **True**.<br>This setting controls whether the page state is compressed. Zipping greatly reduces the memory being taken up by page state in the session object.<br>The following is a snippet of the`web.xml`file:<br>`<param-name>org.apache.myfaces.trinidad.COMPRESS_VIEW_STATE</param-name> <param-value>`**`true`**`</param-value>` | There is an additional CPU cost to zipping and unzipping the view state. |
| `DISABLE_CONTENT_COMPRESSION`<br>**Default**: `False` | Slow initial load of pages. | In production environments, make sure you remove the `DISABLE_CONTENT_COMPRESSION` parameter from the `web.xml` file or set it to `FALSE`. By default, style classes that are rendered are compressed to reduce page size.<br>The following is a snippet of the`web.xml`file:<br>`<param-name>org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION</param-name> <param-value>`**`false`**`</param-value>` | None. |

## 16.3.2 Tuning Process Measurement

Process Analytics uses measurement events to sample the process and publish measurements to registered consumers. In 12*c* (12.2.1.2), these measurements can be enabled by setting the `DisableAnalytics` parameter to `False` in the BPM Enterprise Manager's Analytics Configuration MBean.

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

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

Measurement events are published on the JMS Topic: MeasurementTopic, and consumed by registered Action MDBs. To tune JMS for Measurements, consider changing the parameters listed in Table 16-3, as needed, in a high volume environment:

**Table 16-3 Essential JMS Resource Tuning for BPM**

| JMS Resource | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `dist_MeasurementTopic_auto`<br>**Default**: `Forwarding Policy Replicated` | A distributed measurement topic in a cluster installation is configured by default with `FORWARDING POLICY REPLICATED` even though this is not the best performance option for BPM analytics. | Change the **Forwarding Policy** for this parameter to `PARTITIONED`.<br>This parameter can be altered in the WebLogic console. You can find it from the front page with the following options: **JMS Modules** -> **BPMJMSModule** -> **dist_MeasurementTopic_auto**. You will need to restart all SOA BPM cluster nodes for the changes to take effect. | A distributed topic with a `Partitioned` policy generally outperforms the `FORWARDING POLICY REPLICATED`.<br>For more information on distributed topics versus other topic types, see Supported Topic Types in *Oracle Fusion Middleware Developing Message-Driven Beans for Oracle WebLogic Server*.<br>For more information on partitioned and replicated forwarding policies, see Configuring Partitioned Distributed Topics in *Oracle Fusion Middleware Administering JMS Resources for Oracle WebLogic Server*. |

**Table 16-3 (Cont.) Essential JMS Resource Tuning for BPM**

| JMS Resource | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `MeasurementTopicConnectionFactory`<br>**Default**: `Send Timeout 200000` | You have a high volume environment and you are receiving frequent resource allocation exceptions from message producers.<br>For more information, see Defining a Send Timeout on Connection Factories in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*. | Increase the **Send Timeout** for this parameter to `240000` in a high volume environment.<br>The numerical value represents the maximum length of time in milliseconds.<br>This parameter can be altered in the WebLogic console. You can find it from the front page with the following options: **JMS Modules** -> **BPMJMSModule** --> **MeasurementTopicConnectionFactory** --> **Default Delivery**. | You may create a message backlog that consumes memory and resources. |
| `MeasurementQuota`<br>**Defaults**: `Message Maximum 1000000` and `Bytes Maximum 800000000` | Measurement messages cannot be published and fails with `javax.jms.ResourceAllocationException` thrown. | Set the **Message Maximum** and **Bytes Maximum** for this parameter equal to the amount of system memory available after you have accounted for the rest of your application load.<br>`MeasurementQuota` attributes can be altered in the WebLogic console. You can find it from the front page with the following options: **JMS Modules** -> **BPMJMSModule** -> **MeasurementQuota**. | Increasing this value consumes more memory. Message delivery may still fail if the aggregate size of messages pushed to the consumer is larger than the current protocol's maximum message size.<br>For more information about measurement quotas, see Tuning for Large Messages in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*. |
| `BPMJMSServer`<br>**Default**: `MessageBuffer size 100000` | The JMS server is frequently writing message bodies to disk. | Increase the Message Buffer Size for a given `BPMJMSServer`.<br>Note that the `BPMJMSServer` uses Paging File and `JMSFileStore`.<br>This parameter can be altered in the WebLogic console. You can find it from the front page with the following options: **JMS Servers_auto**_*number*. | The JMS server will use more memory. |

# 17
# Tuning Oracle Human Workflow

You can tune Oracle Human Workflow to optimize its performance in handling various aspects of human interaction with a business process.



## 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 any process. The Human Workflow service consists of a number of services that handle the various aspects of human interaction with a business process.

For more information, see Using the Human Workflow Service Component in *Oracle Fusion Middleware Developing SOA Applications with Oracle SOA Suite*.

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

## 17.2 Tuning Human Workflow

You can tune Oracle Human Workflow to optimize its performance in handling the various aspects of human interaction with a business process.

The suggestions presented here are all applicable to API usage.

**Table 17-1 Essential Human Workflow Tuning Strategies**

| Name | Description | Recommendation |
|---|---|---|
| Minimize Client Response Time | Since workflow client applications are interactive, it is important to have good response time at the client.<br>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. | Review your performance metrics to determine how response time can be improved. |
| Choose the Right Workflow Service Client | Remote client is the best option in terms of performance in most cases. If the client is running in the same JVM as the workflow services (soa-infra application), the API calls are optimized so that there is no remote method invocation (RMI) involved. If the client is on a different JVM, then RMI is used, which can impact performance due to the serialization and deserialization of data between the API methods.<br>SOAP client is preferred for standardization (based on web services). There are additional performance considerations compared to the remote method invocation (RMI) used in the remote client. Additional processing is performed by the web services technology stack, which causes the marshalling and unmarshalling of API method arguments between XML. | If the client application is based on Java EE technology, then consider which client should be used based on your use case scenarios.<br>Note that if the client application is based on .Net technologies, then only the SOAP workflow services can be used. |
| Narrow Qualifying Tasks Using Precise Filters | When a task list is retrieved, the query should be as precise as possible so the maximum filtering can be done at the database level. | Use precise filters to improve response time. |
| Retrieve Subset of Qualifying Tasks (Paging) | The query API has paging parameters that control the number of qualifying rows returned to the user and the start row. | Decrease the `startRow` and `endRow` parameters to values that may limit the number of returned records. This decreases the query time, the application process time, and the amount of data returned to client. |
| Fetch Only the Information That Is Needed for a Qualifying Task | Typically only some of the payload fields are needed for displaying the task list. | When you use the `queryTask` service, consider reducing the amount of optional information retrieved for each task returned in the list.<br>In rare cases where the entire payload is needed, then the payload information can be requested. |

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

| Name | Description | Recommendation |
|---|---|---|
| Reduce the Number of Return Query Columns | When you use the `queryTask` service, consider reducing the number of query columns to improve the SQL time. | Try to use the common columns as they are the most likely indexed columns. This allows the SQL to execute faster. |
| Use the Aggregate API for Charting Task Statistics | Sometimes it is necessary to display charts or statistics to summarize task information. | Consider using the new aggregate APIs to compute the statistics at the database level rather than fetching all the tasks by using the query API and computing the statistics at the client layer. |
| Use the Count API Methods for Counting the Number of Tasks | Sometimes it is only necessary to count how many tasks exist that match certain criteria. | Call the `countTasks` API method, which returns only the number of matching tasks. |
| Create Indexes On Demand for Flexfields | The workflow schema table WFTASK contains several flexfield attribute columns that can be used for storing task payload values in the workflow schema. Because there are numerous columns, and their use is optional, the installed schema does not contain indexes for these columns. | Create indexes on these columns in certain cases where certain mapped flexfield columns are frequently used in query predicates. |
| Use the `doesTaskExist` Method | Sometimes it is necessary to check whether a task exists that matches a particular query criteria. | Consider using the `doesTaskExist` method instead of the default `countTasks`method.<br>The `doesTaskExist` method performs an optimized query that checks if any rows exist that match the specified criteria. This method may achieve better results than calling the `countTasks` method. |

# 17.3 Using Other Tuning Strategies

You can consider using the following strategies to further improve performance.

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

## 17.3.1 Improving Server Performance

Server performance essentially determines the scalability of the system under heavily loaded conditions. In Tuning Human Workflow, strategy *Minimize Client Task Response Time* lists several ways in which client response times can be minimized by fetching the right of amount of information and reducing the potential performance impact that is associated with querying. These techniques also reduce the database

and service logic performance impacts on the server and can improve server performance. In addition, a few other configuration changes can be made to improve server performance:

**Table 17-2 Essential server performance tuning strategies**

| Name | Description | Recommendation |
|---|---|---|
| Archive Completed Instances Periodically | The database scalability of a system is largely dependent on the amount of data in the system. Since business processes and workflows are temporal in nature, once they are processed, they are not queried frequently. | Consider using an archival scheme to periodically move completed instances to another system that can be used to query historical data. Archival should be done carefully to avoid orphan task instances. |
| Select the Appropriate Workflow Callback Functionality | The workflow callback functionality can be used to query or update external systems after any significant workflow event, such as assignment or task completion. | Ensure that there are sufficient resources to update the external system after the task is completed instead of after every workflow event.<br>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 that is associated with a BPEL callback since the callback method is executed in the same thread. |
| Minimize Performance Impacts from Notification | Notifications are useful for alerting users that they have a task to execute. In environments where most approvals happen through email, actionable notifications are especially useful. This also implies that there is not much load in terms of worklist usage. | Minimize the notification to alert a user only when a task is assigned instead of sending out notifications for each workflow event.<br>Also consider making the notifications secure, in which case only a link to the task is sent in the notification and not the task content itself. |
| Deploy Clustered Nodes | All workflow instances and state information are stored in the dehydration database. Workflow services are stateless, which means they can be used concurrently on a cluster of nodes. | When performance is critical and a highly scalable system is needed, a clustered environment can be used for supporting workflow. |

## 17.3.2 Completing Workflows Faster

The time it takes for a workflow to complete depends on the routing type that is specified for the workflow. The workflow functionality provides some options that can be used to decrease the amount of time it takes to complete workflows.

**Table 17-3 Essential workflow completion tuning strategies**

| Name | Description | Recommendation |
|---|---|---|
| Use Workflow Reports to Monitor Progress | Several workflow reports (and corresponding views) are available that can make monitoring and proactive problem fixing easier. | By checking the unattended tasks report, you can assign tasks that have been in the queue for a long time to specific users.<br>By monitoring cycle time and other statistics, you can add staff to groups that are overloaded or take a longer time to complete their tasks. |
| Specify Escalation Rules | To ensure that tasks do not get stuck at any user, you can specify escalation rules. For example, you can move a task to a manager if a certain amount of time passes without any action being taken on the task. If the task must be escalated to some other user based on alternative routing logic, custom escalation rules can also be plugged in. | By specifying proper escalation rules, you can reduce workflow completion times. |
| Specify User and Group Rules for Automated Assignment | Rules can help significantly reduce workflow waiting time, which results in faster workflow completion. | Instead of manually reassigning tasks to other users or members of a group, you can use user and group rules to perform automated reassignment. This ensures that workflows get timely attention. |
| Use Task Views to Prioritize Work | A user's inbox can contain tasks of various types with various due dates. The user has to manually shift through the tasks or sort them to find out which one the user should work on next. | By creating task views where tasks are filtered based on due dates or priority, users can get their work prioritized automatically so they can focus on completing their tasks instead of wasting their time on deciding which tasks to work on. |

## 17.3.3 Tuning the Identity Provider

The workflow service uses information from the identity provider in constructing the SQL query to determine the tasks that qualify for a user based on the role or group membership. The identity provider is also queried for determining role information to determine privileges of a user when fetching the details of a task and determining what actions the user can perform on a task. There are a few ways to speed up requests made to the identity provider.

- Set the search base in the identity configuration file to the nodes as specific as possible. Ideally, you should populate workflow-related groups under a single node to minimize traversal for search and lookup. This is not always possible; for example, you may need to use existing groups and grant membership to groups located in other nodes. If it is possible to specify filters that can narrow down the nodes to be searched, then you should specify them in the identity configuration file.

- Index all critical attributes such as **dn** and **cn** in the identity provider. This ensures that when a search or a lookup is done, only a subset of the nodes are traversed instead of a full tree traversal.
- Use an identity provider that supports caching. Not all LDAP providers support caching but Oracle Internet Directory supports caching, which can make lookup and search queries faster.
- If you use Oracle Internet Directory as the Identity Provider, ensure that you run the `oidstats.sql` to gather latest statistics on the database after the data shape has changed.

## 17.3.4 Tuning the Database

The Human Workflow schema is shipped with several indexes defined on the most important columns. Based on the type of request, different SQL queries are generated to fetch the task list for a user. The database optimizer evaluates the cost of different plan alternatives (for example, full table scan, access table by index) and decides on a plan that is lower in cost. For the optimizer to work correctly, the index statistics should be current at all times. As with any database usage, it is important to make sure that 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 Tuning Database Parameters.

# 18
# Tuning Oracle Business Activity Monitoring

You can tune Oracle Business Activity Monitoring (BAM) to optimize its performance in monitoring business services and processes in the enterprise.



## 18.1 About Oracle Business Activity Monitoring

Oracle Business Activity Monitoring (BAM) provides the tools for monitoring business services and processes in the enterprise.

It allows correlation of market indicators to the actual business process and to change business processes quickly or taking corrective actions if the business environment changes.

Oracle BAM also provides the necessary tools and runtime services for creating dashboards that display real-time data inflow and define rules to send alerts under specified conditions.

For information on how to monitor your BAM installation's performance, see Monitoring Oracle BAM Performance in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM*.

## 18.2 Tuning BAM Server Parameters

You can improve performance of the BAM server by following certain tuning recommendations.

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

- The Weblogic Server. See *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*.
- Metadata Service. See Tuning Oracle Metadata Service.
- Coherence. See *Oracle Fusion Middleware Administering Oracle Coherence*.
- ADF. See Tuning Oracle Application Development Framework (ADF).
- Database Settings. See Tuning Database Parameters.

- Java Virtual Machines (JVMs). See Tuning Java Virtual Machines (JVM) in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*.
- Oracle Platform Security Service. See Tuning Oracle Fusion Middleware Security .

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

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

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

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

**Table 18-1 Essential BAM Server Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `Max connections to BAMDatasource`<br>**Default**: `50` | The number of concurrent connection to the data source consistently reaches max limit. | Increase count as long as the performance continues to scale linearly.<br>This is set at the WebLogic level. The value can be determined mainly based on the number of processors, listening source threading model and max concurrent request settings of embedded servers. | Increasing the count will most likely increase the system resources usage. |
| `Viewset Expiry Time`<br>**Default**: `180` seconds | Viewsets are lingering after the DC connection is lost. | Decrease the expiry time value so that viewsets do not linger.<br>See Monitoring Viewsets in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM* for information on how to find and modify this parameter. | None. |
| `DiagnosticLevel`<br>**Default**: `Info` | You need granular diagnostic logs to identify a problem.<br>OR<br>Your system is running fine and you do not need detailed logs. | Keeping the default of `INFO` will help performance.<br>For more information on using the BAM Diagnostic Framework, see Using the BAM Diagnostic Framework in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM*. | If your system slows down, you do not have detailed logs to identify a problem. |

**Table 18-1 (Cont.) Essential BAM Server Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `ASM (Automatic Server Migration)`<br>**Default**: `WSM` | You want to migrate a SOA Suite installation with BAM to High Availability. Because BAM is a real-time system, you should enable ASM. | ASM is used so High Availability can occur faster than WSM. Given that BAM is a real-time system, ASM is required for BAM HA. | None. |
| `JVM heap size`<br>**Default**: `-Xms768m -Xmx1536m` | Oracle BAM is running slowly and an out-of-memory exceptions occur. | Increase the heap size to 2 GB. Use the following command with the -Xms2048m and -Xmx2048m arguments:<br>`setenv USER_MEM_ARGS "-Xms2048m -Xmx2048m -XX:PermSize=256m -XX:MaxPermSize=768m"` | Increasing the JVM heap size for BAM could affect other SOA components.<br>For more heap size tuning tips, see Tuning Tips for Heap Sizes in *Oracle Fusion Middleware Tuning Performance of Oracle WebLogic Server*. |

# 18.3 Other Tuning Strategies

If Oracle BAM is running more slowly than expected, you can try other tuning strategies.

- Creating an Index Column
- Tuning Loggers
- Tuning Continuous Query Service

## 18.3.1 Creating an Index Column

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

## 18.3.2 Tuning Loggers

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

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

```
oracle.beam.common.alertsengine
oracle.beam.server.service.alertsengine
oracle.beam.Common
oracle.beam.cqservice
oracle.beam.composer
com.oracle.beam
oracle.beam.datacontrol
oracle.beam.datacontrol.management
oracle.beam.server.service.ems
oracle.beam.messaging
```

```
oracle.beam.server.service.persistence
oracle.beam.server.service.reportcache
oracle.beam.security
oracle.beam.mbean
oracle.beam.shared
oracle.beam.server
oracle.beam.impexp.t2p
oracle.beam
```

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

## 18.3.3 Tuning Continuous Query Service

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

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

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

**Table 18-2 Tuning the Continuous Query Service**

| Parameter | Problem | Tuning Recommendation |
|---|---|---|
| `Data Object type`<br>**Default**: None | You have arbitrarily designated simple data objects as stream, archived stream, and archived relation, and are not sure what to do. | Categorize your data objects as stream if you do not care about historical data.<br>See Data Object Types in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM* for detailed descriptions of each data object type and relation. |
| `Data Object purging`<br>**Default**: Disabled | By default, data object retention is not set. Many rows in the data object cause performance issues. | Customer can set Data Object retention in the Data Object **Retention** tab to specify how many days they want to keep the data in a Data Object. When the specified number of days has elapsed, the data rows are automatically purged.<br>See Setting Data Retention in a Data Object in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM* for information on how to find and change this setting. |

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

| Parameter | Problem | Tuning Recommendation |
|---|---|---|
| `Replay for Archived Stream Data Objects` | Data parsing is slow for archived stream data objects. | Specify a smaller **Replay Unit** or a lower **Replay Amount** to reduce the amount of past data retained in memory. This reduces the time and memory to parse data retrieved from the database. |
| `Time Window on Input Streams` | You have chosen to turn an Active Data query into a continuous query and are receiving out-of-memory exceptions. | Decrease the time window size on the Active Data stream. This restricts the amount of memory the window uses to store elements.<br>To get an idea of how much the window size affects memory usage, consider a scenario where the Window Size = 1 hour (RANGE 1 hour) and the event size = 100 bytes. If the event rate is 1000 events / second, then the window will contain 1000 * 3600 events when it is full. The memory consumed is 1000 * 3600 * 100 bytes = ~340 MB.<br>See Enabling Active Data in a View in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM* for information on how to configure the window size on an active data view. |
| `Active Data Collapsing Interval`<br>**Default**: Unchecked | You have checked the box for Active Data Collapsing to make data aggregation active. You want more frequent snapshots or need to free up memory. | Define a smaller **Interval** to make the view update more frequently and to reduce the amount of aggregated data stored in memory.<br>You can maximize your memory usage by taking note of the evaluation interval, the event size, and the event rate. Given the following values:<br>Interval: Every 5 minutes<br>Event Size: 100 bytes<br>Event Rate: 1000 events/second<br>The maximum size of the aggregated view is 5 * 60 * 1000 = 300,000 events = ~28 MB.<br>See Using Active Data in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM* for information on finding the **Active Data Collapsing** setting. |

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

| Parameter | Problem | Tuning Recommendation |
|---|---|---|
| `Slow Changing Dimension for Data Object Dimension Tables`<br>**Default**: `Unchecked` | Continuous queries on dimension tables are slow and consuming memory. | Check this property to activate it. This indicates that the data in this dimension table changes infrequently.<br>For information on specifying slow-changing dimensions for a data object, see Specifying Slow-Changing Dimensions for a Data Object in *Oracle Fusion Middleware Monitoring Business Activity with Oracle BAM*. |
| `Query Type`<br>**Default**: `SQL` | You are experiencing out-of-memory exceptions and most of your queries are continuous. | Use schedule query (SQL) where you do not expect frequent output. This saves memory because SQL involves JDBC resources while CQL stores data in memory. |

# 19
# Tuning Oracle Service Bus

You can tune Oracle Service Bus (OSB) to optimize its performance in providing connectivity, routing, mediation, management, and also some process orchestration capabilities between two or more applications.



## 19.1 About Oracle Service Bus

Within a SOA framework, Oracle Service Bus (OSB) provides connectivity, routing, mediation, management, and also some process orchestration capabilities.

The design philosophy for OSB is to be a high performance and stateless (non-persistent state) intermediary between two or more applications. However, given the diversity in scale and functionality of SOA implementations, OSB applications are subject to a large variety of usage patterns, message sizes, and QOS requirements.

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

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

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

## 19.2 Tuning OSB Parameters

Oracle Service Bus performance largely depends on the performance of the other components.

The following components affect OSB performance:

- WebLogic Server
- Coherence

- Adapters

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

- Tuning Oracle Service Bus with Work Managers
- Tuning OSB Operation Settings

## 19.2.1 Tuning Oracle Service Bus with Work Managers

Starting in 12*c* (12.2.1), Oracle Service Bus can be tuned by several Oracle WebLogic Server Work Managers.

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

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

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

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

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

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

## 19.2.2 Tuning OSB Operation Settings

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

**Table 19-1 Essential OSB Operation Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| Monitoring and Alerting<br>**Default**: `Disabled` | The Monitoring and Alerting framework is designed to have minimal impact on performance, but all of these processes have performance impacts.<br>In general, the more monitoring rules and pipeline actions you have defined, the larger the performance impact. | Keep the default of `Disabled` at the OSB level. Most settings can be defined globally or per service.<br>The settings for monitoring and alerting can be configured in the Enterprise Manager Administrator Console.<br>Note that monitoring must be enabled for SLA alerts but not for Pipeline alerts. | Disabling these processes to improve performance means you are sacrificing certain metrics and alerts that could help you troubleshoot issues in the future.<br>For more information on the OSB Monitoring Framework, see Introduction to the Oracle Service Bus Monitoring Framework in *Oracle Fusion Middleware Administering Oracle Service Bus*. |
| Tracing<br>**Default**: Disabled | If you have large message sizes and high throughput scenarios, tracing may be slowing your system down. | Leave tracing disabled to improve performance.<br>For more information, see How to Enable or Disable Tracing in *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*. | If disabled, you lose metrics.<br>Tracing prints the entire message context, including headers and message body. This is an extremely useful feature both in a development and production environment for debugging, diagnosing, and troubleshooting problems involving message flows in one or more proxy services. |

**Table 19-1 (Cont.) Essential OSB Operation Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `com.bea.wli.sb.pipeline.RouterRuntimeCache.size`<br><br>**Default**: 100 | You may have one of the following issues:<br>Proxy services are accessed slowly.This means you want to store more proxy services in the static portion of the OSB cache for pipeline service runtime metadata. The proxy services stored here are never garbage-collected, meaning they are accessed faster.<br>OR<br>You are seeing a lot of cache misses in DMS dumps. | If you want to include more proxy services in the static cache, increase this value as long as there is sufficient memory for runtime data processing for large number of proxy services.<br>If you are seeing cache misses in DMS dumps, decrease this value.<br>This system property caps the number of proxy services in the static portion of the OSB cache for pipeline service runtime metadata. These services never get garbage collected.<br>You set the size of this value in the `setDomainEnv.sh` file as an extra java argument, as follows:<br>`-Dcom.bea.wli.sb.pipeline.RouterRuntimeCache.size={size}`<br>For example, if you want to set this value to 3000, you would write:<br>`EXTRA_JAVA_PROPERTIES="-Dcom.bea.wli.sb.pipeline.RouterRuntimeCache.size=3000 ${EXTRA_JAVA_PROPERTIES}"` | Increasing this value decreases the time it takes to make initial calls to the proxy server. It can also preload the cache when a configuration session is committed. However, while caching proxy services helps reduce compilation costs, it also increases memory consumption.<br>Decreasing this value may means you free up memory, but making initial calls to the proxy server may take longer. |

**Table 19-1 (Cont.) Essential OSB Operation Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| `reorderJsonAsPerXmlSchema`<br>**Default**: False | JSON input to REST service may not be ordered as expected by the schema definition.<br>When converting from JSON to XML, OSB runtime uses the order in which JSON name or value appear to construct the corresponding XML element. While well-formed, this format is not valid according to XML schema. | Set this parameter to `True` by running the REST wizard and checking the box on the first page.<br>Checking this option makes the REST service reorder the input JSON so that the response from the external REST endpoint can be ordered as per the valid schema definition. | Using this option adds significant performance overhead. |

# 19.3 Using Other Tuning Strategies

After you have performed the recommended modifications, you can make additional changes that are specific to your deployment.

Consider carefully whether the additional tuning recommendations are appropriate for your environment.

- Tuning Resequencer in OSB
- Considering Design Time for Proxy Applications
- Tuning XQuery
- Tuning Poller-based Transports

## 19.3.1 Tuning Resequencer in OSB

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

You can fine-tune the Resequencer by setting the properties listed in Table 19-2 using the Global operational settings page in the OSB EM console:

**Table 19-2 Essential Resequencer Tuning**

| Parameter | Problem | Tuning Recommendation | Trade-offs |
|---|---|---|---|
| ResequencerMaxGroupsLocked<br>**Default**: 4 groups | This parameter defines the maximum number of message groups that can be locked by resequencer locker threads for parallel processing. The locked groups can then use worker threads to process their respective messages.<br>If message processing is being delayed, identify which of the following situations is true:<br>• Incoming messages belong to many groups.<br>• There are many messages and they belong to fewer groups. | If you have many groups with a small number of messages each, increase this parameter's value. Resequencer will lock more groups in one iteration.<br>If you have a few groups with many messages, decrease this value. Resequencer will lock less number of groups for processing. | Increasing the `MaxGroupsLocked` value may result in locking more groups than there are available worker threads. This could result in groups getting blocked while waiting for the availability of the worker threads for message processing.<br>Decreasing the default value may result in under utilization of resources. |
| ResequencerLockerThreadSleep<br>**Default**: 10 seconds | The resequencer locker thread queries the database to lock groups for parallel processing. When no groups are available, the locker thread *sleeps* for the configured amount of time specified by this parameter.<br>If you have either of the following situations, this parameter needs tuning:<br>• You have a high number of messages and processing time between database queries is slow.<br>• You have few messages but frequent database queries. | Decrease this parameter value if you have a high number of messages to reduce the lag time during processing.<br>If Resequencer locker threads are making frequent database round trips even though you do not have many incoming messages, increase this value. | If the sleep time is too short, there may not be enough worker threads available to process incoming messages of the locked groups. Too many database queries will also cause slow performance.<br>If the time interval between incoming messages is already long, configuring a higher value is not beneficial. |
| DeleteMessageAfterComplete<br>**Default**: `True` | The resequencer database is low on space. If you changed this parameter's value to false, processed messages remain in the resequencer database and slow down database inquiries. | Keep the default value of `True` to delete message after successful execution. This frees up database space. | You do not have a detailed history of processed messages. |

## 19.3.2 Considering Design Time for Proxy Applications

Consider the design configurations described in Table 19-3 for proxy applications based on your OSB usage and use case scenarios:

**Table 19-3 Tuning Design Time for Proxy Application**

| Strategy | Description | Recommendations |
|---|---|---|
| Avoid creating many OSB context variables that are used once within another XQuery | Context variables created by 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 by using a `FLWOR` expression. Intermediate values can be created by using *let* statements. | Avoiding redundant context variable creation eliminates overheads that are associated with internal data format conversions. This benefit has to be balanced against visibility of the code and reuse of the variables. |
| Transform contents of a context variable such as `$body`. | Transforming the contents of a context variable could be time-consuming. | Use a Replace action to complete the transformation in a single step.<br>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 (for example, $body/Order) and selecting *Replace entire node*.<br>Leaving the XPath field blank eliminates an extra XQuery evaluation. |
| Specify a special XPath. | A general XPath like `$body/Order` must be evaluated by the XQuery engine before the primary transformation resource is executed. OSB treats `$body/*[1]` as a special XPath that can be evaluated without invoking the XQuery engine. | Use `$body/*[1]` to represent the contents of `$body` as an input to a Transformation (XQuery / XSLT) resource.<br>This is faster than specifying an absolute path pointing to the child of `$body`. |

**Table 19-3 (Cont.) Tuning Design Time for Proxy Application**

| Strategy | Description | Recommendations |
|---|---|---|
| Enable streaming for pure content-based routing scenarios. | OSB leverages the partial parsing capabilities of the XQuery engine when streaming is used in conjunction with indexed XPaths.<br>See Tuning XQuery for additional details. | Enabling streaming means that the payload is parsed and processed only to the field referred to in the XPath. Streaming also eliminates the overhead that is associated with parsing and serialization of XmlBeans.<br>**Trade-offs**: If the payload is accessed a large number of times for reading multiple fields, the gains from streaming can be negated. If all fields read are located in a single subsection of the XML document, a hybrid approach provides the best performance.<br>The output of a transformation is stored in a compressed buffer format either in memory or on disk. Therefore, streaming should be avoided when running out of memory is not a concern. |
| Set the appropriate QOS level and transaction settings. | OSB can invoke a back end HTTP service asynchronously if the QOS is *Best- Effort*. Asynchronous invocation allows OSB to scale better with long running back-end services. It also allows Publish over HTTP to be truly fire-and-forget. | Do not set XA or Exactly-Once unless the reliability level required is once and only once and it is possible to use the setting. If the client is a HTTP client it is not possible to use this setting. If OSB initiates a transaction, it is possible to replace XA with LLR to achieve the same level of reliability. |
| Disable or delete all log actions. | Log actions add an I/O overhead. Logging also involves an XQuery evaluation, which can be expensive. Writing to a single device (resource or directory) can also result in lock contentions. | Disable or delete all log actions. |

## 19.3.3 Tuning XQuery

OSB uses XQuery and XPath extensively for various actions like Assign, Replace, and Routing Table. The following XML structure (`$body`) is used to explain XQuery and XPath tuning concepts:

```
<soap-env:Body>
<Order>
<CtrlArea>
<CustName>Mary</CustName>
</CtrlArea>
<ItemList>
<Item name="ACE_Car" >20000 </Item>
<Item name=" Ext_Warranty" >1500</Item>
…. a large number of items
</ItemList>
```

```
<Summary>
<Total>70000</Total>
<Status>Shipped</Status>
<Shipping>My Shipping Firm </Shipping>
</Summary>
</Order>
</soap-env:Body>
```

You can use the tuning strategies listed in Table 19-4 to tune XQuery.

**Table 19-4 XQuery Tuning Strategies**

| Strategy | Description | Recommendations |
|---|---|---|
| Avoid the use of double front slashes (***//***) in XPaths. | ***//***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 when applicable. | Indexing helps your system process only what is needed. When indexing, only the top part of the document is processed by the XQuery engine. | Index an XPath by adding **[1]**after each node of the path.<br>For example, the XPath `$body/Order/CtrlArea/CustName` implies returning all instances `Order` under `$body` and all instances of `CtrlArea` under `Order`. The entire document has to be read to correctly process the above XPath.<br>But if you know that there is a single instance of `Order` under `$body` and a single instance of `CtrlArea` under `Order`, you can index the above XPath by rewriting it as `$body/Order[1]/CtrlArea[1]/CustName[1]`. This only returns the first instances of the child nodes.<br>**Note**: Do not index when you need a whole array of nodes returned. Indexing only returns the first item node of the array. |
| Extract frequently used parts of a large XML document as intermediate variables within a FLWOR expression. | An intermediate variable can be used to store the common context for multiple values. | Using intermediate variables consumes more memory but reduces redundant XPath processing. |

**Table 19-4 (Cont.) XQuery Tuning Strategies**

| Strategy | Description | Recommendations |
|---|---|---|
| Use a hybrid approach for read-only scenarios with streaming. | If the payload is accessed a large number of times for reading multiple fields, The gains from streaming can be negated. If all fields read are located in a single subsection of the XML document, a hybrid approach provides the best performance. | Enable streaming at the proxy level and assigning the relevant subsection to a context variable. The individual fields can then be accessed from this context variable.<br>The fields `Total` and `Status` can be retrieved by using three `Assign` actions:<br>`Assign "$body/Order[1]/ Summary[1]" to "foo"`<br>`Assign "$foo/Total" to "total"`<br>`Assign "$foo/Status" to "total"` |

> **Note:**
>
> Pipelines enabled for content streaming should use *XQuery 1.0*. Using *XQuery 2004* does work, but incurs a significant performance overhead, as there are *on-the-fly* conversions that happen to and from XQuery 1.0 engine. There is a design-time warning to that effect.

## 19.3.4 Tuning Poller-based Transports

Latency and throughput of poller-based transports depends on the frequency with which a source is polled and the number of files and messages read per polling sweep.

- Setting the Polling Interval
- Setting Read Limit

### 19.3.4.1 Setting the 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 *Oracle Fusion Middleware Developing Services with Oracle Service Bus* |
| FTP Transports | 60 seconds | FTP Transport Configuration Page in *Oracle Fusion Middleware Developing Services with Oracle Service Bus* |

| Polling Intervals | Default Interval | Additional Information |
|---|---|---|
| MQ Transport | 1000 milliseconds | MQ Transport Configuration Page in *Oracle Fusion Middleware Developing Services with Oracle Service Bus* |
| SFTP Transport | 60 seconds | SFTP Transport Configuration Page in *Oracle Fusion Middleware Developing Services with Oracle Service Bus* |
| JCA Transport | 60 seconds | JCA Transport Configuration Page in *Oracle Fusion Middleware Developing Services with Oracle Service Bus* |

## 19.3.4.2 Setting Read Limit

The read limit determines the number of files or messages that are read per polling sweep. You can tune it with the information in Table 19-5.

For more information, see Using the File Transport in *Oracle Fusion Middleware Developing Services with Oracle Service Bus*.

**Table 19-5 Essential Read Limit Tuning**

| Parameter | Symptoms if not properly tuned | Tuning Recommendation | Performance Trade-offs |
|---|---|---|---|
| Read Limit<br>**Default**: 10 for File and FTP transports | Excessive memory use or high memory use due to a large number of files read into memory simultaneously. | Set this value to the desired concurrency. It can be set to 0 to specify no limit.<br>The read limit determines the number of files or messages that are read per polling sweep. | Setting the **Read Limit** to a high value and the **Polling Interval** to a small value may result in a large number of messages being simultaneously read into memory. If the message size is large, this can lead to an out-of-memory error . |

# 20

# Tuning Oracle Enterprise Scheduler Service

You can tune Oracle Enterprise Scheduler Service (ESS) to optimize its performance in enabling scheduling and running jobs.



## 20.1 About Enterprise Scheduler Service

Oracle Enterprise Scheduler enables scheduling and running jobs within a particular time frame, or workshift, by using rules to create work assignments.

Oracle Enterprise Manager Fusion Applications Control allows you to define, control and manage Oracle Enterprise Scheduler job metadata, including job definitions, job requests, job sets (a collection of job requests), incompatibilities (job definitions and job sets that cannot run at the same time for a given application) and schedules governing the execution of job requests.

For more information, see Introduction to Administering Oracle Enterprise Scheduler in *Oracle Fusion Middleware Administering Oracle Enterprise Scheduler*.

## 20.2 Tuning Enterprise Scheduler Service Parameters

You can tune the enterprise scheduler service parameters for optimal performance.

Table 20-1 describes the enterprise scheduler service tuning parameters.

`Maximum Poll Interval` is a dispatcher parameter that applies to the Oracle Enterprise Scheduler request dispatcher. The request dispatcher manages requests that are awaiting their scheduled execution. The request processor handles the job requests once they have dispatched.

`Thread Count` is a processor tuning parameter that applies to the Oracle Enterprise Scheduler request processor. The request processor manages job requests whose scheduled execution time has arrived, and are ready to execute.

**Table 20-1 Essential Enterprise Scheduler Service Tuning**

| Name | Symptoms | Recommendations | Trade-offs |
|---|---|---|---|
| `Maximum Poll Interval`<br>**Default**: 15 seconds | A high number of requests whose execution time has been reached and remain in `WAIT` state for an extended time. | If there is an excess of waiting requests that are eligible to be dispatched and processor threads are available, decrease this value . | Lowering the value increases CPU usage and database activity.<br>Increasing the value may delay the dispatching of requests that are ready for processing. |
| `Thread Count`<br>**Default**: 25 | A high number of requests in `READY` state that are otherwise available for processing. | If there is a build up of requests that are ready to be executed and the increase system resource usage is acceptable, increase this value.<br>Lower the value to reduce the amount of system resources used for request processing. | Increasing this value increases CPU usage, memory usage, and database activity.<br>Lowering this value may result in a build up and potentially delay processing of requests. |

21

# Tuning Oracle Business Intelligence Performance

You can tune Oracle Business Intelligence to optimize its performance in collecting, presenting, and delivering data.

- 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.
- Tuning Oracle BI Server Query Performance
  You can improve query performance by tuning and indexing underlying databases, by using aggregate tables, query caching.
- Tuning Oracle BI Server Query Cache Performance
  You can configure the Oracle BI Server to maintain a local, disk-based cache of query result sets (query cache).
- Tuning Oracle BI Web Client Performance
  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.

## 21.1 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.

## 21.2 Tuning Oracle BI Server Query Performance

You can improve query performance by tuning and indexing underlying databases, by using aggregate tables, query caching.

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 by 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.

# 21.3 Tuning Oracle BI Server Query Cache Performance

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*.

# 21.4 Tuning Oracle BI Web Client Performance

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 11*g* 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 WebCenter Components

The Oracle WebCenter components need to be tuned for optimal performance.

This part describes configuring Oracle WebCenter components to improve performance. It contains the following topic:

- Tuning Oracle WebCenter Portal
  You can tune Oracle WebCenter Portal to optimize its performance as a deployed application.

22

# Tuning Oracle WebCenter Portal

You can tune Oracle WebCenter Portal to optimize its performance as a deployed application.

- About Oracle WebCenter Portal
  Oracle WebCenter Portal helps companies to build enterprise-scale intranet and extranet portals that provide a foundation for the next-generation user experience (UX) with Oracle Fusion Middleware and Oracle Fusion Applications.
- Basic Tuning Considerations
  Tuning considerations apply to most WebCenter Portal application deployment scenarios.
- Tuning Configuration for WebCenter Portal
  You can tune configuration parameters to improve the performance of WebCenter Portal.
- Tuning Tools and Services Configuration
  You can tune the performance of tools and services used by WebCenter Portal.
- Tuning Identity Store Configuration
  Performance-related configurations may be required for specific environments.
- Tuning Portlet Configuration
  You can tune the performance of portlets in WebCenter Portal.

## 22.1 About Oracle WebCenter Portal

Oracle WebCenter Portal helps companies to build enterprise-scale intranet and extranet portals that provide a foundation for the next-generation user experience (UX) with Oracle Fusion Middleware and Oracle Fusion Applications.

Portals built with Oracle WebCenter Portal commonly support thousands of users who create, update, and access content and data from multiple back-end sources. Oracle WebCenter Portal delivers intuitive user experiences by leveraging the best UX capabilities from a significant portfolio of leading portal products and related technologies. From the user's perspective, the integration is seamless.

Business users can easily assemble new portals or composite applications by using Portal Composer and a page editor that includes a library of prebuilt reusable components. They can enhance user experience by wiring components together on the page, configuring content personalization, enabling the use of integrated social tools, and creating data visualizations.

For more information about Oracle WebCenter Portal, see:

- *Oracle Fusion Middleware Using Oracle WebCenter Portal*
- *Oracle Fusion Middleware Building Portals with Oracle WebCenter Portal*
- *Oracle Fusion Middleware Administering Oracle WebCenter Portal*

- *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*

# 22.2 Basic Tuning Considerations

Tuning considerations apply to most WebCenter Portal application deployment scenarios.

It is highly recommended that you review these configurations and implement those that meet your particular usage requirements.

- Setting System Limit
- Setting JDBC Data Source
- Setting JRockit Virtual Machine (JVM) Arguments
- Using Content Compression to Reduce Downloads

## 22.2.1 Setting System Limit

To run WebCenter Portal 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.

## 22.2.2 Setting JDBC Data Source

To determine the correct setting for the JDBC data source, use the Oracle WebLogic Server Administration Console to monitor the running system database connection usage as described Configuring JDBC Data Sources. If the *Waiting for Connection Failure* rate is noticeably higher, and the *Active Connections Current Count* is close to reaching the maximum capacity, then consider increasing the capacity to avoid potential database connection contention.

However, if the *Active Connections Current Count* is routinely lower than the maximum capacity, consider reducing the capacity to save memory.

For more information, see Configuring Connection Pool Features in *Oracle Fusion Middleware Administering JDBC Data Sources for Oracle WebLogic Server*.

The following data source settings are WebCenter Portal defaults for data sources **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-
frequency-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-
connection>
        <statement-timeout>-1</statement-timeout>
        <pinned-to-thread>false</pinned-to-thread>
    </jdbc-connection-pool-params>
```

For information on how to edit MDS data source settings, see Tuning Data Source Connection Pools in *Oracle Fusion Middleware Administering JDBC Data Sources for Oracle WebLogic Server*.

## 22.2.3 Setting JRockit Virtual Machine (JVM) Arguments

JVM arguments are set in the `setDomainEnv.sh` file on Unix operating systems and the `setDomainEnv.cmd` file on Windows operating systems. The `setDomainEnv` file is located in the *`domain_dir`*`/bin` directory.

- **WebLogic Server production mode:** When WebCenter Portal 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 by 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 an out-of-memory error occurs frequently, then increase the heap size as appropriate to your server's available physical memory.

  For more information, see Tuning Java Virtual Machines (JVMs) 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
  ```

## 22.2.4 Using Content Compression to Reduce Downloads

If clients connect to your server using relatively slow connections, that is, by using modems or VPN from remote locations, consider compressing the content before it downloads to the client. While content compression increases the load on the server, the client's download experience is much improved.

> **Note:**
>
> Beginning with release 11.1.1.8.0, WebCenter Portal is preconfigured with an ADF caching filter, which automatically sets up caching for static resources and do compression. This preconfigured ADF caching filter is available only for use with WebLogic Server.

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 the `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 the `mod_deflate` module, refer to: `http://httpd.apache.org/docs/2.0/mod/mod_deflate.html`.

# 22.3 Tuning Configuration for WebCenter Portal

You can tune configuration parameters to improve the performance of WebCenter Portal.

- Setting a Session Timeout for WebCenter Portal
- Setting MDS Cache Size and Purge Rate
- Configuring Concurrency Management

## 22.3.1 Setting a Session Timeout for WebCenter Portal

The default session timeout for the WebCenter Portal application is 45 minutes. Administrators can customize the session time to suit their installation. For details see Specifying Session Timeout Settings in *Oracle Fusion Middleware Using Oracle WebCenter Portal*.

## 22.3.2 Setting MDS Cache Size and Purge Rate

If you encounter the any of the following conditions, then you can increase the MDS cache size in the `adf-config.xml` file. The default MDS cache size is 100 MB.

- Error message `JOC region full`
- Frequent MDS database access after the page is warmed up
- Retained memory by ADF application is close to the `max-size-kb`

Post deployment, modify these properties through the System MBeans Browser. For more information, see Changing MDS Configuration Attributes for Deployed Applications in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

The following is a sample snippet of the `adf-config.xml` file:

```
<cache-config>
<max-size-kb>150000</max-size-kb>
</cache-config>
```

Purging MDS data improves MDS queries. If your portal site changes frequently, you may want to purge old MDS data more often, by reducing the time between purges.

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, old versions of metadata are automatically purged every hour, that is, the `auto-purge seconds-to-live` parameter is set to 3600 seconds (as shown in the example below).

> **Note:**
>
> Each purge incurs CPU usage in the database. Do not purge too often (for example, every 5 or 10 minutes) because the database CPU impact might outweigh the performance gains from the purge.

If excessive metadata is accumulated and each purge is very expensive, reduce this interval in the `adf-config.xml` file.

By default, there is no `auto-purge` entry in the `adf-config.xml` file. Use the following sample snippet of the `adf-config.xml` file to modify auto-purge:

```
<mdsC:adf-mds-config version="11.1.1.000">
    <mds-config xmlns="http://xmlns.oracle.com/mds/config">
      <persistence-config>
        <metadata-namePortal>
           ...
```

```
      </metadata-namespace>
      <auto-purge seconds-to-live="3600"/>
    </persistence-config>
```

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='[ServerName]',olderThan=10
)
```

## 22.3.3 Configuring Concurrency Management

Concurrency management includes global settings that impact the entire WebCenter Portal and the 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 code snippet describes the format of the global, service, and resource entries in the `adf-config.xml` file:

```
<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 the `connections.xml` file or the `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 the attributes except `service` and `resource` are optional, and therefore, for example, the following tags are valid:

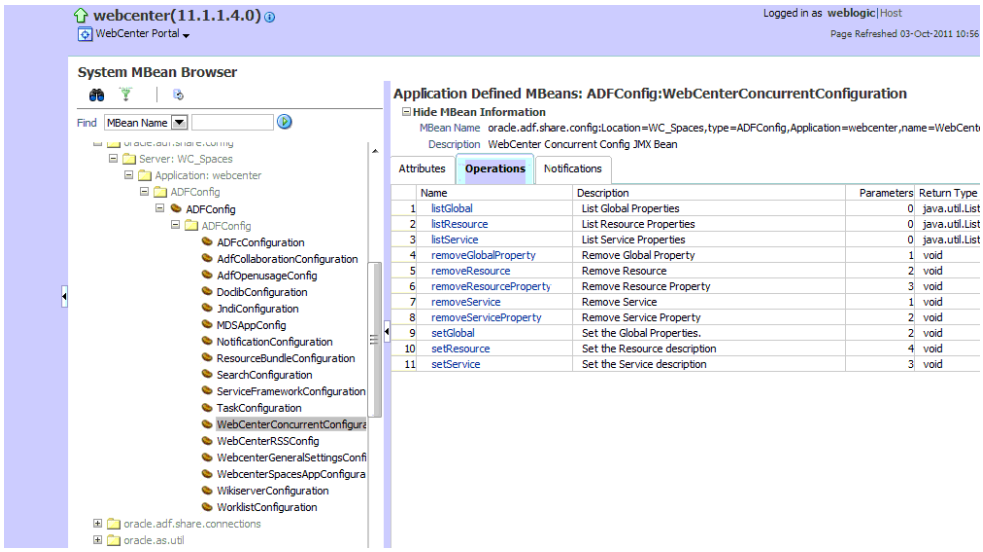
```
<global queueSize="20"/>
    <resource service="foo" resource="bar" timeoutMaxPeriod="5s"/>
```

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, see Accessing the System MBean Browser in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

1. In System MBean Browser, navigate to:

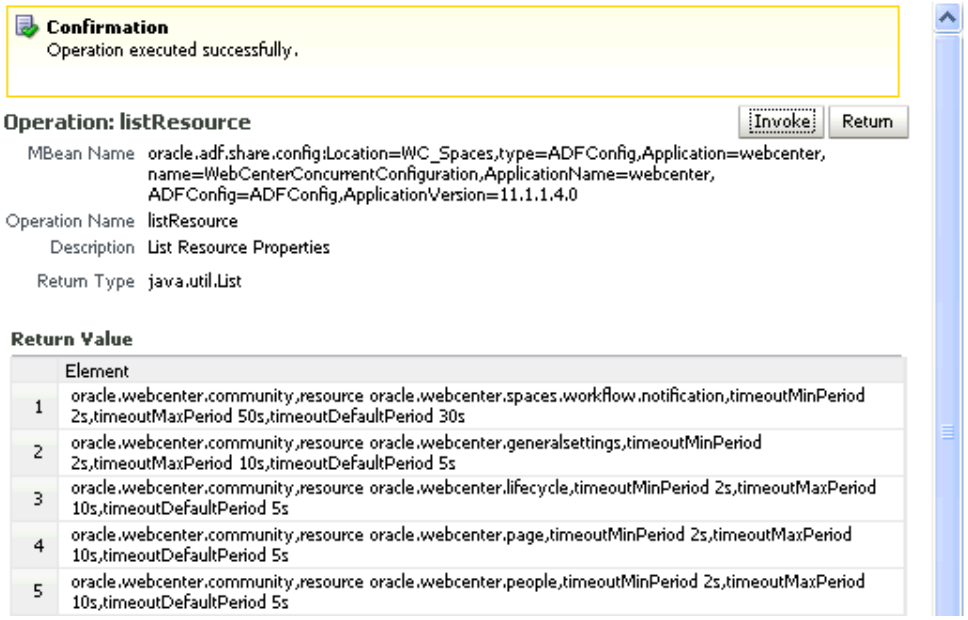
   **Application Defined MBeans** -> **oracle.adf.share.config** -> **Server: (your server name)** -> **Application: (your application name)** ->**ADFConfig** -> **ADFConfig (bean)** -> **ADFConfig** -> **WebCenterConcurrentConfiguration** -> **Operations** -> **listResource**

**Figure 22-1 System MBean Browser - WebCenterConcurrentConfiguration**



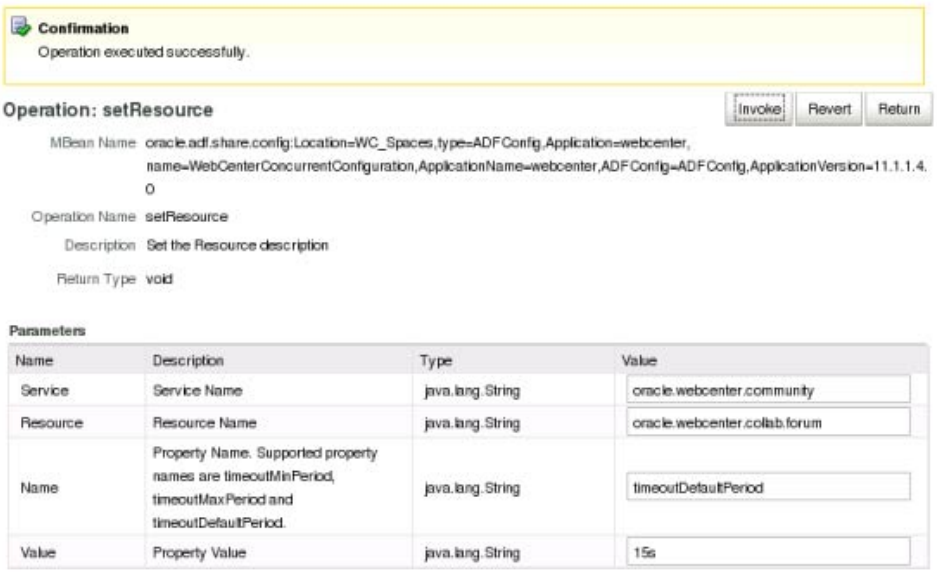

2. To view the current concurrency settings, select **listResource**, and then click **Invoke** (Figure 22-2).

**Figure 22-2 System MBean Browser - listResource**




3. To change a setting, select **setResource**, enter the resource details, and then click **Invoke** (Figure 22-3).

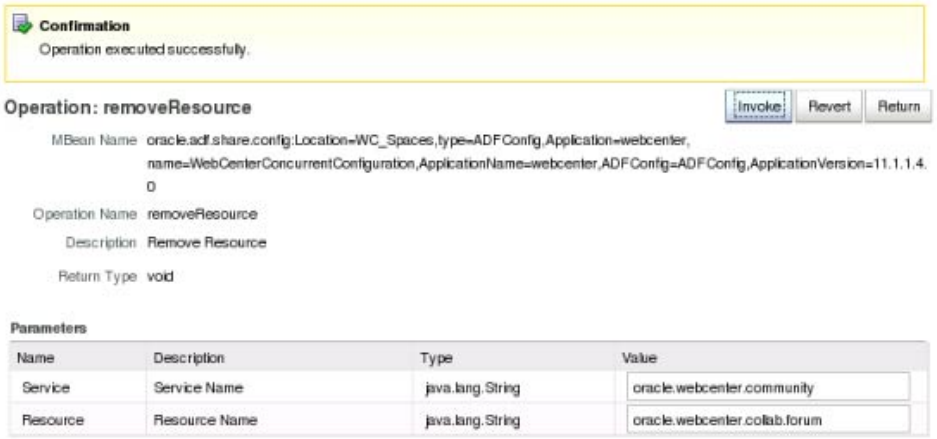
**Figure 22-3 System MBean Browser - setResource**




Take care to enter the correct values for **service**, **resource**, **name** and **value**.

> **Note:**
>
> If the resource parameter that you are attempting to modify already has a [**value**] setting, you must remove the setting first by invoking the [**removeResource**] operation (Figure 22-4).

**Figure 22-4 System MBean Browser - removeResource**




4. To save changes, navigate to **Application Defined MBeans: ADFConfig:ADFConfig** -> **save**, and click **Invoke**.

# 22.4 Tuning Tools and Services Configuration

You can tune the performance of tools and services used by WebCenter Portal.

For information about how to tune and improve the performance of back-end servers, for example, mail servers, BPEL servers, content servers, and so on, refer to the appropriate product documentation for each server.

- Tuning Performance of Announcements
- Tuning Performance of Discussions
- Tuning Performance Instant Messaging and Presence
- Tuning Performance of Mail
- Tuning Performance of Personal Events
- Tuning Performance of RSS News Feeds
- Tuning Performance of Searches
- Tuning Policy Store Parameters

## 22.4.1 Tuning Performance of Announcements

To manage overall resource usage for the announcements, 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 by using WLST. For details, see:

- Modifying Discussions Server Connection Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.
- Modifying Discussions Server Connection Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

The following is a sample code snippet of the `connections.xml` file 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.ForumConnectionFactory"/
>
          <StringRefAddr addrType="connection.time.out">
             <Contents>5</Contents>
          </StringRefAddr>
       </RefAddresses>
</Reference>
```

## 22.4.2 Tuning Performance of Discussions

To manage the overall resource usage for discussions, 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 by using WLST. For details, see:

- Modifying Discussions Server Connection Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*
- Modifying Discussions Server Connection Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*

The following is a sample snippet of the `connections.xml` file:

```
<Reference name="Jive-7777"
className="oracle.adf.mbean.share.connection.webcenter.forum.ForumConnection">
         <Factory
className="oracle.adf.mbean.share.connection.webcenter.forum.ForumConnectionFactory"/
>
         <RefAddresses>
            <StringRefAddr addrType="forum.url">
               <Contents>http://[machine]:[port]/owc_discussions_5520</Contents>
               <StringRefAddr addrType="connection.time.out">
               <Contents>5</Contents>
            </StringRefAddr>
         </RefAddresses>
</Reference>
```

## 22.4.3 Tuning Performance Instant Messaging and Presence

To manage the overall resource usage for instant messaging and presence, 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 by using WLST. For details, see:

- Modifying Instant Messaging and Presence Connections Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.
- Modifying Instant Messaging and Presence Connections Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

The following is a sample code snippet of the `connections.xml` file 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>
```

## 22.4.4 Tuning Performance of Mail

To manage the overall resource usage for mail, 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 by using WLST. For details, see:

- Modifying Mail Server Connection Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*
- Modifying Mail Server Connection Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*

The following is a sample code snippet of the `connections.xml` file 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>
```

## 22.4.5 Tuning Performance of Personal Events

To manage the overall resource usage for personal events, 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 by using WLST. For details, see:

- Modifying Event Server Connection Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*

- Modifying Event Server Connection Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*

The following is a sample code snippet of the `connections.xml` file to change the default timeout to `5` seconds:

```
<Reference name="MSExchange-my-
pc"className="oracle.adf.mbean.share.connection.webcenter.calendar.PersonalEventConne
ction">
  <Factory
className="oracle.adf.mbean.share.connection.webcenter.calendar.PersonalEventConnecti
onFactory"/>
  <StringRefAddr addrType="eventservice.connection.timeout">
    <Contents>5</Contents>
  </StringRefAddr>
  <StringRefAddr addrType="eventservice.cache.expiration.time">
    <Contents>5</Contents>
  </StringRefAddr>
 </RefAddresses>
</Reference>
```

## 22.4.6 Tuning Performance of RSS News Feeds

To manage the overall resource usage for RSS news feeds, 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 the `adf-config.xml`file:

```
<rssC:adf-rss-config>
       <rssC:RefreshSecs>3600</rssC:RefreshSecs>
       <rssC:TimeoutSecs>3</rssC:TimeoutSecs>
       <rssC:Configured>true</rssC:Configured>
</rssC:adf-rss-config>
```

## 22.4.7 Tuning Performance of Searches

To manage the 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 all search results (in ms).

Post deployment, modify timeout properties through Fusion Middleware Control or by using WLST. For details, see:

- Modifying Oracle SES Connection Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.
- Modifying Oracle SES Connection Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

The following is a sample snippet of the `adf-config.xml`file:

```
<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>
```

## 22.4.8 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 OPSS PDP Service Tuning Parameters. Depending on your use case scenarios, performance of WebCenter Portal 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 portals in your WebCenter Portal deployment.

> **Note:**
>
> Only modify this parameter if your WebCenter Portal deployment expects to have more than 3000 active portals.

- Set `oracle.security.jps.policystore.policy.cache.size` to 5 times the expected number of portals.

> **Note:**
>
> Always refer to your own use case scenarios before you modify the policy store parameters. For more information, see *Oracle Fusion Middleware Administering Web Services* before tuning any security parameters.

# 22.5 Tuning Identity Store Configuration

Performance-related configurations may be required for specific environments.

- Tuning the Identity Store when Using SSL
- Tuning Performance when Using OVD
- Tuning Performance when Using Active Directory

## 22.5.1 Tuning the Identity Store when Using SSL

When you configure an identity store for WebCenter Portal, 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.

## 22.5.2 Tuning Performance when Using OVD

For Oracle Virtual Directory (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>
```

### 22.5.3 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=pr
imaryGroupID" />
        <property value="sAMAccountName" name="username.attr"/>
        <property value="sAMAccountName" name="user.login.attr"/>
      </serviceInstance>
```

Profiles query for all these attributes and there is no default mapping for these attributes in the Active Directory provider. An out-of-the-box Active Directory installation doesn't have any mapping corresponding to `DATE_OF_HIRE`, `DATE_OF_BIRTH`.

Note that these 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.

## 22.6 Tuning Portlet Configuration

You can tune the performance of portlets in WebCenter Portal.

- Tuning Performance of the Portlet Client
- Enabling Java Object Cache for WSRP Producers
- Customizing the Container Runtime Environment Options
- Tuning Performance of Oracle PDK-Java Producers
- Setting WSRP Attribute for Portlet-served Resources
- Setting WSRP Attribute for Resources Not Served by the Portlet
- Tuning Performance of OmniPortlet

### 22.6.1 Tuning Performance of the Portlet Client

Several tuning options are available for Portlet Client.

- Configuring Supported Locales
- Configuring Portlet Cache Size
- Configuring Portlet Timeout

### 22.6.1.1 Configuring Supported Locales

To manage the 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 and ready-to-use. You can remove the locales that are unnecessary for your application.

If you must modify these properties, post deployment, you must edit the `adf-config.xml` file manually. See Editing adf-config.xml in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

The following is a sample snippet of the `adf-config.xml` file:

```
<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>
```

### 22.6.1.2 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 the `adf-config.xml` file manually.

For more information, see How to Edit Portlet Client Configuration in *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*

### 22.6.1.3 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 the `adf-config.xml` file manually. See Editing adf-config.xml in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

The following is a sample snippet of the `adf-config.xml` file:

```
<adf-portlet-config>
     ....
  <defaultTimeout>5</defaultTimeout>
  <minimumTimeout>2</minimumTimeout>
  <maximumTimeout>300</maximumTimeout>
</adf-portlet-config>
```

## 22.6.2 Enabling Java Object Cache for WSRP Producers

For Portal Framework applications, Oracle recommends that you enable the Java Object Cache (JOC) for WSRP producers so that objects written to the persistent store are cached.

For more information, see Enabling Java Object Cache for Database Persistence Store Access in *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

## 22.6.3 Customizing the Container Runtime Environment Options

Customizing container runtime options can improve overall performance.

For more information, see How to Customize the Runtime Environment for JSR 286 Portlets in *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

- Suppressing Optimistic Rendering for WSRP Portlets
- Setting Portlet Container Runtime Options
- Excluding Request Attributes for Portlets

### 22.6.3.1 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 in the `portlet.xml` file to `true`. For example:

```
com.oracle.portlet.suppressWsrpOptimisticRender=true
```

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 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.

For more information, see How to Customize the Runtime Environment for JSR 286 Portlets in *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

### 22.6.3.2 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 that 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.

For more information, see How to Customize the Runtime Environment for JSR 286 Portlets in *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

### 22.6.3.3 Excluding Request Attributes for Portlets

The `excludedActionScopeRequestAttributes` is a multivalued, Portlet container runtime property, where each value is a regular expression.

If you use the `javax.portlet.actionScopedRequestAttributes` container runtime option with a portlet, it is possible to optimize the request attributes that are stored between portlet lifecycles by using the `com.oracle.portlet.excludedActionScopeRequestAttributes` container runtime option.

Any request attributes that are unnecessary to store between lifecycles can be indicated to increase performance.

For more information, see How to Customize the Runtime Environment for JSR 286 Portlets in *Oracle Fusion Middleware Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

## 22.6.4 Tuning Performance of Oracle PDK-Java Producers

To manage the 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 by using WLST. For details, see:

- Editing WSRP Producer Registration Details Using Fusion Middleware Control in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.
- Editing Producer Registration Details Using WLST in *Oracle Fusion Middleware Administering Oracle WebCenter Portal*.

The following is a sample snippet of the `connections.xml` file:

```
<webproducerconnection producerName="wc-WebClipping" urlConnection="wc-WebClipping-
urlconn" timeout="10000" establishSession="true" mapUser="false"/>
```

## 22.6.5 Setting WSRP Attribute for Portlet-served Resources

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.

## 22.6.6 Setting WSRP Attribute for Resources Not Served by the Portlet

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.

## 22.6.7 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 Administering Oracle WebCenter Portal*.
- Editing Producer Registration Details Using WLST in *Oracle Fusion Middleware Administering 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"/>
```