Sun Java System Access Manager 7.1 Performance Tuning and Troubleshooting Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–4673–20
February 22, 2008

080227@19028

Contents

Preface 9

Part I Basic Performance Tuning 15

1 Introduction to Access Manager Tuning 17
Before You Begin 17
Tuning Access Manager and Other Components 18

2 Access Manager Tuning Scripts 19
Overview of the Access Manager Tuning Scripts 19
Tuning Scripts for Windows Systems 21
Tuning Modes 21
Running an Access Manager Tuning Script 22
▼ To run a tuning script: 22
Access Manager `amtune-env` File Parameters 23
Access Manager Tuning Parameters 24
Installation Environment Tuning Parameters 28
Web Server 7.0 Tuning Parameters 32
Application Server 8 Tuning Parameters 33

3 Directory Server Tuning 37
Directory Server Tuning Parameters 37
Directory Server Tuning Scripts 38
Directory Server Indexes 39
Running the Directory Server Tuning Script in REVIEW Mode 39
Applying the Tuning Changes to Directory Server 40

4 Distributed Authentication UI Server Tuning 41
Copying the Tuning Scripts 41
Tuning the Operating System 42
Tuning a Distributed Authentication UI Server Web Container 42
▼ To tune a Distributed Authentication UI server web container: 43
Improving Performance for the Default Application User 43
▼ To improve performance for the Distributed Authentication UI server default user: 43

5 Tuning Considerations 45
Operating System (OS) Considerations 45
Solaris OS 45
Linux OS 46
Third-Party Web Containers 49
IBM WebSphere Application Server 49
BEA WebLogic Server 51

Part II Troubleshooting Performance Issues 55

6 Best Practices for Performance Tuning and Testing 57
Avoiding Common Performance Testing and Tuning Mistakes 57
Using a Systematic Approach to Performance Tuning 58
Constructing the System 58
Automated Performance Tuning 58
Related Systems Tuning 58
Baseline Modular Performance Testing 59
Advanced Performance Tuning 67
Targeted Performance Testing 67

7 Advanced Performance Tuning 69
Tuning the LDAP Connection Pool and LDAP Configurations 69
To Tune the User Authentication LDAP Configuration 70
To Tune the Data Store LDAP Configuration 70
To Tune the Access Manager Configuration Store and SMS LDAP Configuration 70
Resolving Memory Issues 71

To Tune the Notification Threadpool Size 73
To Tune the Purge Delay Settings 73
Performance Tuning Questions and Answers 74
More Resources 77

Part III Appendix 79

A Known Issues and Workarounds 81
Memory Grows or Leaks 81
WS 6.1 bundled with JES 4 fail to start 81
System Responds Too Slowly 81
Application Server where Access Manager is deployed throws "cannot create thread" error 82
Directory Server 5.2 hangs and shows high CPU usage when deleting entries 82
Throughput performance of AM is significantly slower when it is deployed on WebLogic or WebSphere Application Server. 83
"OutOfMemoryError" When Access Manager is deployed in WebLogic or WebSphere Application Server 83
Server Hangs and Does Not Respond 83
Access Manager Server hangs during session failover 84
Server hangs when processing request between the load balancer and the Access Manager server. 85
Access Manager server hangs when Sun Java System Directory Server restarts 85
Access Manager unable to recover after a crash or watch dog restart under heavy load 86
jaxrpc getAttributes throws SSOException 86
Sun Java System Web Server hangs while handling a large number of images files 86
Access Manager Web Policy Agent hangs 86
Access Manager server hangs when multiple clients point to one Access Manager server instance 87
System hangs when Access Manager clientsdk.jar and Access Manager server are in the same JVM instance 87
Server Crashes 87
Access Manager web container crashes with "StackOverFlowError" errors 87
Apache Web Agent 2.2 on Linux crashes 88
Access Manager crashes in SSL mode 88
Customized JSP page causes Web Server to crash 88

Application Server or Web Server crashes under a heavy load 88

B Error Messages 91

Index 101

Tables

TABLE 2–1 Access Manager Tuning Scripts 20
TABLE 2–2 Access Manager Tuning Parameters 24
TABLE 2–3 Installation Environment Tuning Parameters 29
TABLE 2–4 Web Server 7.0 Tuning Parameters 33
TABLE 2–5 Application Server 8 Web Container Tuning Parameters 34
TABLE 3–1 Directory Server Tuning Parameters 38

Preface

The *Sun Java™ System Access Manager 7.1 Performance Tuning and Troubleshooting Guide* describes how to tune Access Manager 7.1 and its related components, including the Solaris or Linux operating system, Access Manager web container, and Directory Server, for optimal performance.

Access Manager is a component of the Sun Java Enterprise System (Java ES), a set of software components that provide services needed to support enterprise applications distributed across a network or Internet environment.

Who Should Use This Book

This book is primarily intended for system and network administrators who are tuning Access Manager 7.1 and its related components.

Before You Read This Book

You should be familiar with the following components and concepts:

- Access Manager technical concepts, as described in the *Sun Java System Access Manager 7.1 Technical Overview*.
- Deployment platform: Solaris™ or Linux operating system.
- Access Manager Web container: Sun Java System Application Server, Sun Java System Web Server, BEA WebLogic Server, or IBM WebSphere Application Server.
- Technical concepts: Lightweight Directory Access Protocol (LDAP), Java technology, JavaServer Pages™ (JSP) technology, HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), and eXtensible Markup Language (XML).

How This Book Is Organized

This book is organized in three Parts:

Part I Basic Performance Tuning

- Chapter 1, "Introduction to Access Manager Tuning," is an introduction to Access Manager performance tuning.
- Chapter 2, "Access Manager Tuning Scripts," describes how to run the Access Manager tuning scripts.
- Chapter 3, "Directory Server Tuning," describes how to tune Sun Java System Directory Server.
- Chapter 4, "Distributed Authentication UI Server Tuning," describes tuning considerations for a Distributed Authentication UI server.
- Chapter 5, "Tuning Considerations," provides considerations for the Solaris OS, Linux OS, and third-party web containers, including IBM WebSphere Application Server and BEA WebLogic Server.

Part II Troubleshooting Performance Issues

- Chapter 6, "Best Practices for Performance Tuning and Testing," describes recommended procedures for obtaining optimum tuning and testing results.
- Chapter 7, "Advanced Performance Tuning," provides performance troubleshooting tips.

Part III Appendixes

- Appendix A, "Known Issues and Workarounds" provides solutions to known performance problems.
- Appendix B, "Error Messages" describes common performance error messages and their solutions.

Related Books

Related documentation is available as follows:

- "Access Manager Core Documentation" on page 10
- "Related Sun Java Enterprise System Documentation" on page 12

Access Manager Core Documentation

The following table describes the Access Manager documentation set, which is available on the following web site:

http://docs.sun.com/coll/1292.2

TABLE P–1 Access Manager 7.1 Documentation Set

Title	Description
Sun Java System Access Manager 7.1 Documentation Center	Provides links to commonly referenced information in the Access Manager 7.1 documentation collection.
Sun Java System Access Manager 7.1 Release Notes	Describes new features, problems fixed, installation notes, and known issues and limitations. The Release Notes are updated periodically after the initial release to describe any patches, new features, or problems.
Sun Java System Access Manager 7.1 Technical Overview	Explains basic Access Manager concepts and terminology and provides an overview of how Access Manager components work together to consolidate access control functions and to protect enterprise assets and web-based applications.
Sun Java System Access Manager 7.1 Deployment Planning Guide	Provides planning and deployment solutions for Access Manager based on the solution life cycle.
Sun Java System Access Manager 7.1 Postinstallation Guide	Provides information about configuring Access Manager after installation. Usually, you perform postinstallation tasks only a few times. For example, you might want to deploy an additional instance of Access Manager or configure Access Manager for session failover.
Sun Java System Access Manager 7.1 Administration Guide	Describes various administrative tasks such as realms management, policy management, authentication, and directory management.
Sun Java System Access Manager 7.1 Administration Reference	Provides reference information for the Access Manager command-line interface (CLI), configuration attributes, `AMConfig.properties` attributes, `serverconfig.xml` file attributes, log files, and error codes.
Sun Java System Access Manager 7.1 Federation and SAML Administration Guide	Provides information about Federation Manager based on the Liberty Alliance Project specifications. It includes information on the integrated services based on these specifications, instructions for enabling a Liberty-based environment, and summaries of the application programming interface (API) for extending the framework.
Sun Java System Access Manager 7.1 Developer's Guide	Provides information about customizing Access Manager and integrating its functionality into an organization's current technical infrastructure. It also contains details about the programmatic aspects of the product and its API.
Sun Java System Access Manager 7.1 C API Reference	Provides summaries of data types, structures, and functions that make up the Access Manager public C APIs.

TABLE P–1 Access Manager 7.1 Documentation Set *(Continued)*

Title	Description
Sun Java System Access Manager 7.1 Java API Reference	Provides information about the implementation of Java packages in Access Manager.
Sun Java System Access Manager 7.1 Performance Tuning and Troubleshooting Guide (this guide)	Provides information about how to tune Access Manager and its related components for optimal performance.
Sun Java System Access Manager Policy Agent 2.2 User's Guide	Provides an overview of Policy Agent software, including the web agents and J2EE agents that are currently available. To view the Access Manager Policy Agent 2.2 documentation collection, see: `http://docs.sun.com/coll/1322.1`

Related Sun Java Enterprise System Documentation

The following table provides links to documentation collections for related Java ES products.

TABLE P–2 Related Sun Java Enterprise System Documentation

Product	Link
Sun Java Enterprise System 5	`http://docs.sun.com/coll/1286.2`
Sun Java System Directory Server Enterprise Edition 6	`http://docs.sun.com/coll/1224.1`
Sun Java System Web Server 7	`http://docs.sun.com/coll/1308.3`
Sun Java System Application Server Enterprise Edition 8.2	`http://docs.sun.com/coll/1310.3`
Sun Java System Message Queue 3.7 UR1	`http://docs.sun.com/coll/1307.2`
Sun Java System Web Proxy Server 4.0.4	`http://docs.sun.com/coll/1311.4`

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–3 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–4 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com`and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document.

For example, the title of this book is *Sun Java System Access Manager 7.1 Performance Tuning and Troubleshooting Guide*, and the part number is 819-4673-11.

PART I

Basic Performance Tuning

- Chapter 1: Introduction to Access Manager Tuning
- Chapter 2: Access Manager Tuning Scripts
- Chapter 3: Directory Server Tuning
- Chapter 4: Distributed Authentication UI Server Tuning
- Chapter 5: Tuning Considerations

◆ ◆ ◆ CHAPTER 1

Introduction to Access Manager Tuning

This guide provides performance tuning information for Sun Java™ System Access Manager, including how to run the Access Manager tuning scripts. You can run these scripts to tune Access Manager and its related components.

Before You Begin

Before you use this guide, Access Manager and other Sun Java Enterprise System component products such as Directory Server, Web Server, and Application Server must be installed. For information about installing these products, see the one of the following books:

- *Sun Java Enterprise System 5 Installation Guide for UNIX*
- *Sun Java Enterprise System 5 Installation Guide for Microsoft Windows*

Caution – Tuning Access Manager and its related components is an iterative process that can vary for different deployments. The Access Manager tuning scripts try to apply the optimal tuning parameter settings. However, each deployment is unique and might require further customization to suit specific requirements.

You can run the Access Manager tuning scripts in two modes:

- REVIEW mode (default): The scripts return tuning recommendations in the `amtune` debug log file, but they do not make any actual changes to the deployment.
- CHANGE mode: The scripts make the tuning changes to the deployment that are defined in the `amtune-env` file.

Therefore, it is recommended that you first run a script in REVIEW mode and check the recommended changes in the debug log file. Run a script in CHANGE mode only after you have reviewed the recommended tuning changes that will be applied to your deployment.

Tuning Access Manager and Other Components

This guide includes the following information:

- Chapter 2, "Access Manager Tuning Scripts," describes how to run the Access Manager tuning scripts.
- Chapter 3, "Directory Server Tuning," describes how to tune Sun Java System Directory Server.
- Chapter 4, "Distributed Authentication UI Server Tuning,"Chapter 4, "Distributed Authentication UI Server Tuning," describes tuning considerations for a Distributed Authentication UI server.
- Chapter 5, "Tuning Considerations," provides considerations for the Solaris OS, Linux OS, and third-party web containers, including IBM WebSphere Application Server and BEA WebLogic Server.

◆ ◆ ◆ CHAPTER 2

Access Manager Tuning Scripts

The Sun Java™ System Access Manager 7.1 tuning scripts allow you to tune Access Manager and other components of your deployment, including Sun Java System Directory Server, the web container running Access Manager, and the operating system (OS) kernel and TCP/IP parameters.

This chapter includes the following topics:

- "Overview of the Access Manager Tuning Scripts" on page 19
- "Access Manager `amtune-env` File Parameters" on page 23

Overview of the Access Manager Tuning Scripts

The Access Manager tuning scripts are non-interactive. To run a script, you first edit the parameters in the `amtune-env` configuration file to specify the tuning options you want to set for your specific environment. Then, you run either the `amtune` script, which calls other scripts as needed, or a specific script. For example, you might run only the `amtune-identity` script to tune only Access Manager.

The Access Manager tuning scripts and the `amtune-env` configuration file are installed in the following directory, depending on your platform:

- Solaris systems: *AccessManager-base*`/SUNWam/bin/amtune`
- Linux systems: *AccessManager-base*`/identity/bin/amtune`
- Windows systems: *javaes-install-directory*`\identity\bin\amtune`

AccessManager-base is the Access Manager 7.1 base installation directory. The default base installation directory is `/opt` on Solaris systems and `/opt/sun` on Linux systems.

On Windows systems, the default value for *javaes-install-directory* is `C:\Program Files\Sun\JavaES5`.

The following table describes the tuning scripts that are available in the Access Manager 7.1 release. See also "Tuning Scripts for Windows Systems" on page 21.

TABLE 2–1 Access Manager Tuning Scripts

Script	Description
`amtune`	Wrapper script that calls other scripts based on values in the `amtune-env` file.
`amtune-identity`	Tunes the installed instance of Access Manager. Before you run the `amtune-identity` tuning script, it is recommended that you add the following property set to `false` to the `AMConfig.properties` file: `com.sun.identity.log.resolveHostName=false` A value of `false` minimizes the impact of resolving host names and thus can improve performance. (However, if you want the client machine's hostname to be printed in the `amAuthentication.access` log, set the value to `true`.) This script supports Web Server in JVM 64-bit mode. If Web Server 7.0 is the web container, it determines if Web Server 7.0 is running in 64-bit or 32-bit mode and then calculates the tuning parameters accordingly.
`amtune-os` Not available on Windows systems	Tunes the operating system kernel and TCP/IP parameters for both the Solaris OS and Linux OS. The script determines the OS type from the `uname -s` command. The `amtune-os`will not run if the wrapper `amtune` script is run in a global zone on Solaris 10 or higher.
`amtune-ws7`	Tunes the Sun Java System Web Server 7.0 web container. This script supports Web Server in JVM 64-bit mode. It determines if Web Server 7.0 is running in 64-bit or 32-bit mode and then calculates the tuning parameters accordingly.
`amtune-ws61`	Tunes the Sun Java System Web Server 6.1 2005Q4 SP5 web container.
`amtune-as8`	Tunes the Sun Java System Application Server Enterprise Edition 8.2 Web container.
`amtune-as7`	Tunes the Sun Java System Application Server 7 Web container.
`amtune-prepareDSTuner`	Generates the `amtune-directory` script, which you can use to tune the Directory Server that supports Access Manager. For more information, see Chapter 3, “Directory Server Tuning.”

Tuning Scripts for Windows Systems

Access Manager 7.1 includes tuning scripts for Microsoft Windows systems. Running the tuning scripts on a Windows system is similar to running the scripts on a Solaris system or Linux system, with these differences:

- The Windows scripts are written in Perl and require Active Perl 5.8 to run.
- A script to tune the Windows operating system is not available.
- Before running a script, you must set the `$BASEDIR` parameter in the `amtune-env.pl` file to the Access Manager installation directory.
- If you are tuning Directory Server, after running the `amtune-prepareDSTuner.pl` script, you must copy the `amtune-utils.pl`, `amtune-directory.pl`, and `amtune-samplepasswordfile` files to the Directory Server system.
- Support for zones is not provided.

Tuning Modes

The Access Manager tuning scripts can run in the following modes, as determined by the `AMTUNE_MODE` parameter in the `amtune-env` file.

- REVIEW mode (default). The scripts return tuning recommendations for an Access Manager deployment, but they do not make any actual changes to the environment.
- CHANGE mode. The scripts make all of the tuning modifications that are defined in the `amtune-env` file, except for Directory Server. For information, see Chapter 3, "Directory Server Tuning."

In either mode, the scripts return a list of tuning recommendations to the `amtune` debug log file and the terminal window. The location of the log file is determined by the `com.iplanet.services.debug.directory` parameter in the `AMConfig.properties` file. The default debug directory depends on your platform:

- Solaris systems: `/var/opt/SUNWam/debug`
- Linux and HP-UX systems: `/var/opt/sun/identity/debug`
- Windows systems: *javaes-install-directory*`\identity\debug` where the default value for *javaes-install-directory* is `C:\Program Files\Sun\JavaES5`.

Caution – Tuning is an iterative process that can vary for different deployments. The Access Manager tuning scripts try to apply the optimal tuning parameter settings. However, each deployment is unique and might require further customization to suit specific requirements.

Therefore, it is recommended that you first run a script in REVIEW mode and check the recommended changes in the `amtune` debug log file. Run the scripts in CHANGE mode only after you have reviewed the recommended tuning changes that will be applied to your deployment.

Running an Access Manager Tuning Script

To run a tuning script, use the following syntax:

amtune-script admin_password dirmanager_password [*as8_admin_password*]

The tuning script parameters are:

- *amtune-script* is one of the tuning scripts: `amtune`, `amtune-identity`, `amtune-os`, `amtune-ws61`, `amtune-as7`, `amtune-as8`, or `amtune-prepareDSTuner`.
- *admin_password* is the Access Manager Administrator password.
- *dirmanager_password* is the Directory Manager (`cn=Directory Manager`) password.
- *as8_admin_password* is the Administrator password that is required if you are tuning Application Server 8 (`WEB_CONTAINER=AS8`).

▼ To run a tuning script:

Follow these basic steps to run an Access Manager Tuning script.

1. **Log in as or become superuser (`root`).**
2. **If you have not run the script in REVIEW mode, ensure that `AMTUNE_MODE=REVIEW` (which is the default value) in the `amtune-env` file.**
3. **Edit other parameters in the `amtune-env` file, depending on the components you want to tune and the requirements for your deployment:**

   - "Access Manager Tuning Parameters" on page 24
   - "Installation Environment Tuning Parameters" on page 28
   - "Web Server 7.0 Tuning Parameters" on page 32, if Web Server 7.0 is the web container
   - "Application Server 8 Tuning Parameters" on page 33, if Application Server 8 is the web container

   To tune the Directory Server that supports Access Manager, see Chapter 3, "Directory Server Tuning."

4. **In REVIEW mode, run either the `amtune` script or one of the component scripts.**

5. **Review the tuning recommendations in the debug log file. If needed, make changes to the `amtune-env` file based on this run.**

6. **If you are satisfied with the tuning recommendations from the REVIEW mode run, set `AMTUNE_MODE=CHANGE` in the `amtune-env` file.**

7. **In CHANGE mode, run either the `amtune` script or one of the component scripts.**

   For example, to tune the Solaris OS, run the `amtune-os` script, as follows:

   ```
   # ./amtune-os admin_password dirmanager_password
   ```

   Note – In CHANGE mode, the `amtune` script might need to restart the web container and Access Manager. In some instances, `amtune` might also recommend a system restart.

8. **Check the debug log file for the results of the run.**

Consideration for the `com.iplanet.am.session.purgedelay` Property

This property specifies the number of minutes to delay the purge session operation. The value recommended by `amtune` is 0 or 1, depending upon the Access Manager version you're using. For clients such as Sun Java System Portal Server, a higher value may be needed. You must manually set the property after you run the `amtune` script:

1. In the `AMConfig.properties` file, set the property to the new value. For example:

   ```
   com.iplanet.am.session.purgedelay=5
   ```

2. Restart the Access Manager web container for the new value to take effect.

Access Manager `amtune-env` File Parameters

The `amtune-env` file contains the parameters to define the tuning options for an Access Manager deployment, including:

- “Access Manager Tuning Parameters” on page 24
- “Installation Environment Tuning Parameters” on page 28
- “Web Server 7.0 Tuning Parameters” on page 32
- “Application Server 8 Tuning Parameters” on page 33

For a description of the Directory Server parameters, see Chapter 3, “Directory Server Tuning.”

Access Manager Tuning Parameters

The following table describes the specific parameters for tuning Access Manager.

TABLE 2–2 Access Manager Tuning Parameters

Parameter	Description
AMTUNE_MODE	Sets the tuning mode to one of the following: ■ REVIEW– The scripts return tuning recommendations for an Access Manager deployment but do not make any actual changes to the deployment environment. ■ CHANGE– The scripts make all of the tuning modifications that you have defined in the amtune-env file, except for Directory Server. For more information, see Chapter 3, "Directory Server Tuning." Default: REVIEW
AMTUNE_TUNE_OS	Tunes the operating system kernel and TCP/IP settings. Default: true
AMTUNE_TUNE_DS	Generates a script to tune the Directory Server that supports Access Manager. Default: true
AMTUNE_TUNE_ WEB_CONTAINER	Tunes the Access Manager web container, which can be either Web Server or Application Server. Default: true
AMTUNE_TUNE_IDENTITY	Tunes the installed instance of Access Manager. Default: true
AMTUNE_LOG_LEVEL	Specifies the log level for the output of the run: NONE — No results will be logged or displayed. TERM — Display results on the terminal only. FILE — Display the results and log in the debug log file. Default: FILE

TABLE 2–2 Access Manager Tuning Parameters *(Continued)*

Parameter	Description
AMTUNE_DEBUG_ FILE_PREFIX	Identifies the prefix for the amtune log file. If this parameter is set, all operations performed by the amtune scripts are logged. The location of the log file is determined by the com.iplanet.services.debug.directory parameter in the AMConfig.properties file. If Access Manager is not installed on the server, the debug log file is written to the directory when the tuning scripts exist. For example, if a Distributed Authentication UI server is deployed from a WAR file. Default: amtune

TABLE 2–2 Access Manager Tuning Parameters *(Continued)*

Parameter	Description

AMTUNE_PCT_
MEMORY_TO_USE
	Specifies the percent of available memory used by Access Manager.
	
	Currently, Access Manager can use a maximum of 4 GB, which is the per process address space limit for 32-bit applications.
	
	Access Manager requires a minimum of 256 MB RAM.
	
	When you set AMTUNE_PCT_MEMORY_TO_USE to 100, the maximum space allocated for Access Manager is the minimum between 4 GB and 100% of available RAM.
	
	When you set AMTUNE_PCT_MEMORY_TO_USE to 0, Access Manager is configured to use 256 MB RAM
	
	Default: 75
	
	The following values are derived from this parameter setting:
	■ JVM memory usage - Heap sizes, NewSizes, PermSizes
	■ Thread pool sizes - Web Server RqThrottle, Authentication LDAP connection pool, SM LDAP connection pool, Notification thread pools
	■ Access Manager caches - SDK caches and session caches
	■ Maximum sizes - Maximum number of sessions and maximum number of cache entries
	
	AMConfig.properties File Settings
	
	Notification thread pool settings:
	
	`com.iplanet.am.notification.threadpool.size`
	
	`com.iplanet.am.notification.threadpool.threshold`
	
	SDK cache maximum size setting:
	
	`com.iplanet.am.sdk.cache.maxsize`
	
	Session settings:
	
	`com.iplanet.am.session.httpSession.enabled`
	
	`com.iplanet.am.session.maxSessions`
	
	`com.iplanet.am.session.invalidsessionmaxtime`
	
	`com.iplanet.am.session.purgedelay`

TABLE 2–2 Access Manager Tuning Parameters *(Continued)*

Parameter	Description
AMTUNE_PER_THREAD_ STACK_SIZE_IN_KB	Sets the available stack space per thread in Java (Web container). The per thread stack size is used to tune various thread-related parameters in Access Manager and the Web container. Default: 128 KB **Caution**: Do not change this value unless absolutely necessary.
AMTUNE_PER_THREAD_ STACK_SIZE_IN_KB_64_BIT	Sets the available stack space per thread in Java (Web container) when the script detects Web Server 7.0 is running as a 64-bit process. Default: 512 KB
AMTUNE_MEM_MAX_HEAP_SIZE_RATIO	Specifies the maximum heap size ratio that is used to calculate the maximum and minimum heap sizes. Default: 7/8 **Note**: If you are running the amtune-ws7 script with 64-bit enabled and the system has a large memory, the script displays the current value of AMTUNE_MEM_MAX_HEAP_SIZE_RATIO and the maximum and minimum heap sizes calculated from this value. If these values are sufficient, you do not need to make any changes. However, in some situations, you might need to modify the value of AMTUNE_MEM_MAX_HEAP_SIZE_RATIO.
AMTUNE_MIN_MEMORY_TO_USE_IN_MB AMTUNE_MAX_MEMORY_TO_USE_IN_MB	Specifies the minimum and maximum memory in MB that should not be exceeded. Defaults: 512 and 3584 If Web Server 7.0 is running in a 64-bit process, the AMTUNE_MAX_MEMORY_TO_USE_IN_MB parameter is not used. It is recommended that you use the default values.
AMTUNE_DONT_TOUCH_ SESSION_PARAMETERS	Specifies whether session time-out tuning using the next three parameters is enabled. To enable, set to false. Default: true

TABLE 2–2 Access Manager Tuning Parameters *(Continued)*

Parameter	Description
AMTUNE_SESSION_MAX_SESSION_TIME_IN_MTS	Sets the maximum session time in minutes. Default: 60 However, the default value might be different for your installation. If the session service is registered and customized at the any other level, the tuning will not apply. Setting this parameter to very high or very low values affects the number of active user sessions an Access Manager deployment can support, so this parameter is optional for tuning purposes. To use this parameter, AM_TUNE_DONT_TOUCH_SESSION_PARAMETERS must be set to false.
AMTUNE_SESSION_MAX_IDLE_TIME_IN_MTS	Sets the maximum idle time for a session in minutes. Default: 10 However, the default value might be different for your installation. If the Session service is registered and customized at the any other level, the tuning will not apply. Setting this parameter to very high or very low values affects the number of active user sessions an Access Manager deployment can support, so this parameter is optional for tuning purposes. To use this parameter, AM_TUNE_DONT_TOUCH_SESSION_PARAMETERS must be set to false.
AMTUNE_SESSION_MAX_CACHING_TIME_IN_MTS	Sets the maximum session cache time in minutes. Default: 2 However, the default value might be different for your installation. If the Session service is registered and customized at the any other level, the tuning will not apply. Setting this parameter to very high or very low values affects the number of active use sessions an Access Manager deployment can support, so this parameter is optional for tuning purposes. To use this parameter, AM_TUNE_DONT_TOUCH_SESSION_PARAMETERS must be set to false.

Installation Environment Tuning Parameters

The following table describes the Access Manager installation environment tuning parameters.

Note – The OSTYPE, OSPLATFORM, and HWPLATFORM parameters are used to construct other parameters, so you should not need to change their values.

TABLE 2–3 Installation Environment Tuning Parameters

Parameter	Description
HOSTNAME	Specifies the host name of the system where Access Manager is deployed. If the host name for your environment cannot be obtained using the hostname command, comment the following line: HOSTNAME='/bin/hostname | /bin/cut -f1 -d"."' Then, add a line setting the correct host name. For example: HOSTNAME=*myhost*
DOMAINNAME	Specifies the domain name of the system where Access Manager is deployed. If the domain name for your environment cannot be obtained using the domainname command, comment the following line: DOMAINAME='/bin/domainname' Then, add a line setting the correct domain name. For example: DOMAINNAME=example.com
IS_INSTALL_DIR	Specifies the Access Manager installation directory. Default: blank. The tuning scripts determine the default Access Manager installation directory dynamically by the pkginfo or rpm command. If the pkginfo or rpm command fails, values are /opt/SUNWam on Solaris systems or opt/sun/identity on Linux systems. For an Access Manager WAR file deployment, the value should be blank. The IS_INSTALL_DIR and IS_CONFIG_DIR parameters are then replaced by WAR file deployment setup script.
AMTUNE_BIN_DIR	Specifies the location of the tuning scripts. Set this variable only if the tuning scripts are not installed in the default location. Otherwise, leave it blank. Default: *AccessManager-base*/bin/amtune

TABLE 2–3 Installation Environment Tuning Parameters *(Continued)*

Parameter	Description
WEB_CONTAINER	Specifies the name of the Web container on which Access Manager is deployed: ■ WS7 — Web Server 7.0 ■ WS61 — Web Server 6.1 ■ AS8 — Application Server 8 ■ AS7 — Application Server 7 Default: WS7 Any other value returns a validation error.
CONTAINER_BASE_DIR	Specifies the base directory for the Web container that is running Access Manager. If you installed the Web container in a non-default location, change this value before running amtune. Default values: ■ Web Server 7.0: /opt/SUNWwbsvr7 ■ Web Server 6.1: /opt/SUNWwbsvr ■ Application Server 7: /var/opt/SUNWappserver7 ■ Application Server 8 on Solaris systems /var/opt/SUNWappserver ■ Application Server 8 on Linux systems /var/opt/sun/appserver
WEB_CONTAINER_INSTANCE_NAME	Specifies the instance name of the Access Manager web container. Typically, this value is the host name where Access Manager is deployed. If you have multiple instances for the Web container, this value might be different from the host name, and you must set it to the correct instance name. Defaults: ■ Web Server 6.1 or Web Server 7.0: *hostname* (${HOSTNAME}) ■ Application Server 7: domains/server1 ■ Application Server 8: domains/domain1

TABLE 2–3 Installation Environment Tuning Parameters *(Continued)*

Parameter	Description
IS_INSTANCE_NAME	Specifies the Access Manager instance names. IS_INSTANCE_NAME is used to determine the properties file names for the Access Manager installation. Default: none You can deploy multiple instances of Access Manager on the same machine, but generally, there is one set of properties files for each Access Manager instance, and the instance name is appended to the file names. If there is only one instance of Access Manager on a machine, the instance name is not appended to the file name. For example, there might be a single instance of Access Manager running under the default instance of Web Server. If Access Manager is installed on a machine named `server.example.com`, typically the first instance of Web Server is `https-server.example.com`. The properties files for the first Access Manager instance will not have the instance name appended (for example, `AMConfig.properties`). **Multiple Access Manager Instances** Multiple instances will have different names. For example, if there are three instances of Web Server, the Web Server instances might be: ■ `server.example.com-`*instance1* ■ `server.example.com-`*instance2* ■ `server.example.com-`*instance3* If three instances of Access Manager are deployed (one per web container instance), the primary properties file names for Access Manager (typically, `AMConfig.properties`) might be named as: ■ `AMConfig-`*instance1*`.properties` ■ `AMConfig-`*instance2*`.properties` ■ `AMConfig-`*instance3*`.properties`

TABLE 2–3 Installation Environment Tuning Parameters *(Continued)*

Parameter	Description
IS_INSTANCE_NAME (continued)	You can specify IS_INSTANCE_NAME=*instance1*. The amtune script resolves the properties file names in the following order: 1. AMConfig-IS_INSTANCE_NAME 2. AMConfig-WEB_CONTAINER_INSTANCE_NAME 3. AMConfig.properties The script uses the first available properties file in the list. The amadmin utility should also point to the correct server name. Java option: -Dserver.name=IS_INSTANCE_NAME amtune automatically tries to associate the instance names with the Access Manager properties files using this parameter. Currently, only these files are based on this instance name: ■ AMConfig.properties ■ serverconfig.xml
CONTAINER_INSTANCE_DIR	Specifies the base directory for the Access Manager web container instance. If you have installed the web container in a non-default location, change this value before running amtune. Default values are: Web Server 6.1 or Web Server 7.0: $CONTAINER_BASE_DIR/https-${WEB_CONTAINER_INSTANCE_NAME} Application Server 7 or Application Server 8: $CONTAINER_BASE_DIR/${WEB_CONTAINER_INSTANCE_NAME}

Web Server 7.0 Tuning Parameters

The following table describes the tuning parameters that you can set when you are running Web Server 7.0 as the Access Manager web container.

TABLE 2–4 Web Server 7.0 Tuning Parameters

Parameter	Description
WSADMIN	Specifies the location of the wsadmin utility. Default: Solaris systems: /opt/SUNWwbsvr7/bin/wadm Linux systems: /opt/sun/webserver7/bin/wadm
WSADMIN_USER	Specifies the Web Server 7.0 administrator. Default: admin
WSADMIN_PASSFILE	Specifies the Web Server 7.0 temporary password file used by the wsadmin utility. Default: /tmp/passfile
WSADMIN_HOST	Specifies the Web Server 7.0 admin host name. Default: localhost ($HOSTNAME)
WSADMIN_PORT	Specifies the Web Server 7.0 admin port. Default: 8989
WSADMIN_DIR	Specifies the Web Server 7.0 installation directory.
WSADMIN_SECURE	Specifies whether WSADMIN_PORT is a secure port. "--ssl=true" indicates a secure port. "--ssl=false" indicates the port is not secure. Default: "--ssl=true"
WSADMIN_CONFIG	Specifies the Web Server 7.0 instance name. Default: $WEB_CONTAINER_INSTANCE_NAME
WSADMIN_HTTPLISTENER	Specifies the Web Server 7.0 HTTP listener name. Default: http-listener-1

Application Server 8 Tuning Parameters

The following table describes the tuning parameters that you can set when you are using Application Server 8 as the Access Manager web container.

TABLE 2–5 Application Server 8 Web Container Tuning Parameters

Parameter	Description
ASADMIN	Specifies the Application Server 8 asadmin utility location. Default values: ■ Solaris systems: /opt/SUNWappserver/appserver/bin/asadmin ■ Linux systems: /opt/sun/appserver/bin/asadmin
ASADMIN_USER	Specifies the Application Server 8 administrator user account. Default: admin
ASADMIN_PASSFILE	Specifies the temporary password file location used by the asadmin utility. The amtune-as8 script creates this file and then deletes it after use. Default: /tmp/passfile
ASADMIN_HOST	Specifies the Application Server 8 admin host name. Default: $HOSTNAME
ASADMIN_PORT	Specifies the Application Server 8 admin port. Default: 4849
ASADMIN_DIR	Specifies the Application Server 8 installation directory.
ASADMIN_SECURE	Specifies whether the ASADMIN_PORT is secure: ■ "--secure" specifies the port is secure. ■ Blank specifies that the port is not secure. Default: "--secure"
ASADMIN_TARGET	Specifies whether this Application Server 8 installation is used exclusively for Access Manager and Portal Server. Default: server, indicating that Application Server 8 installation is exclusively used for Access Manager and Portal Server.
ASADMIN_HTTPLISTENER	Specifies the HTTP Application Server 8 listener name. Default: http-listener-1
ASADMIN_INTERACTIVE	Specifies whether Application Server 8 administrator operates interactively. Default: false **Caution**: Do not change this parameter.

TABLE 2–5 Application Server 8 Web Container Tuning Parameters *(Continued)*

Parameter	Description
AMTUNE_WEB_CONTAINER_ JAVA_POLICY	Specifies whether Application Server 8 evaluates Java security descriptors, as specified in the server.policy file. Default: false **Caution**: Do not change this parameter. Evaluating Java security descriptors can add a significant performance overhead.

◆ ◆ ◆ CHAPTER 3

Directory Server Tuning

Sun Java™ System Access Manager 7.1 tuning scripts tune either Sun Java System Directory Server 5.2 2005Q4 or Sun Java System Directory Server Enterprise Edition 6. Access Manager must use an existing Directory Server, either local or remote, in non-exclusive mode. If your deployment has separate Directory Servers for the Access Manager configuration data and users, you must manually tune each Directory Server.

Caution – If you are working with a production Directory Server or a Directory Server that has not been backed up (both the data and configuration), it is recommended that you do not run the `amtune-directory` script in CHANGE mode to apply tuning changes.

After you run the `amtune-directory` script in REVIEW mode, review the tuning recommendations and apply them manually, if they meet your deployment needs.

Also, make sure you back up both your Directory Server data and configuration before you make any changes.

This chapter includes the following topics:

- "Directory Server Tuning Parameters" on page 37
- "Directory Server Tuning Scripts" on page 38

Directory Server Tuning Parameters

The following table describes the Directory Server tuning parameters in the `amtune-env` configuration file.

TABLE 3–1 Directory Server Tuning Parameters

Parameter	Description
AMTUNE_TUNE_DS	Generates a script to tune the Directory Server that supports Access Manager. Default: true
DIRMGR_UID	Specifies the user ID of the Directory Manager. If your deployment uses a user ID other than the default value (cn=Directory Manager), you must set this parameter with that value. Default: cn=Directory Manager
RAM_DISK	Specifies the location of the RAM disk. Default: /tmp
DEFAULT_ORG_ PEOPLE_CONTAINER	Specifies the people container name for the default organization. This parameter is used to tune the LDAP authentication module's search base. It can be useful when there are no sub-organizations in the default organization. If this value is empty (""), tuning is skipped. **Note**: Along with appending the people container to the search base, the search scope will be modified to "OBJECT" level. The default search scope is "SUBTREE". Default: "" (empty)
AMTUNE_LOG_LEVEL	Specifies the log level for the output of the run: NONE — No results will be logged or displayed. TERM — Display results on the terminal only. FILE — Display the results and log in the debug log file. Default: FILE

Directory Server Tuning Scripts

The amtune-prepareDSTuner scripts generates the amtune-directory script, which you can then use to tune Directory Server. This chapter describes:

- "Directory Server Indexes" on page 39
- "Running the Directory Server Tuning Script in REVIEW Mode" on page 39

- "Applying the Tuning Changes to Directory Server" on page 40

Directory Server Indexes

The Directory Server tuning script creates the following indexes if they do not already exist:

- Index for attributes that are used to search for a user to be authenticated
- Indexes for the default Access Manager attributes: `nsroledn`, `memberof`, `iplanet-am-static-group-dn`, `iplanet-am-modifiable-by`, `iplanet-am-user-federation-info-key`, `sunxmlkeyvalue`, `o`, `ou`, `sunPreferredDomain`, `associatedDomain`, and `sunOrganizationAlias`

  These attributes are indexed during installation by using the `index.ldif` file in `/etc/opt/SUNWam/config/ldif` directory on Solaris systems and `/etc/opt/sun/identity/config/ldif` directory on Linux systems. If for some reason, any of these attributes are not indexed, the Directory Server tuning script creates them.

  For more information about indexes, see Appendix A, "Directory Server Considerations," in *Sun Java System Access Manager 7.1 Postinstallation Guide*.

Running the Directory Server Tuning Script in REVIEW Mode

The `amtune` script and `amtune-prepareDSTuner` scripts do not actually tune Directory Server. However, you must run one of these scripts to generate the `amtune-directory` script, which you can then use to tune Directory Server.

1. Log in as or become superuser (`root`).
2. Make sure that the following parameter is set in the `amtune-env` file:

   ```
   AMTUNE_TUNE_DS=true
   ```

3. Run the `amtune` script or `amtune-prepareDSTuner` script. The script generates the following `tar` file:

   current-directory`/amtune-directory.tar`
4. Copy the `amtune-directory.tar` file to a temporary location on the server that is running Directory Server.
5. Untar the `amtune-directory.tar` file in the temporary location.
6. In the `amtune-directory` script, make sure REVIEW mode is set:

   ```
   AMTUNE_MODE="REVIEW"
   ```

7. Set these parameters, if you prefer a value other than the default (`amtune`):

- `DEBUG_FILE_PREFIX` is a prefix that will be included with the timestamp to specify the filename of the log file where the script writes the recommended tuning changes.
- `DB_BACKUP_DIR_PREFIX` is a prefix that will be included with the timestamp to specify the name of the Directory Server backup directory.

8. Run the `amtune-directory` script in REVIEW mode. For example:

   ```
   # ./amtune-directory dirmanager_password
   ```

   The *dirmanager_password* is the Directory Manager password.
9. Review the recommended tuning settings for Directory Server in the debug log file.

   The script creates the log file in the same directory with the tuning scripts.

Applying the Tuning Changes to Directory Server

Caution – If you are working with a production Directory Server or a Directory Server that has not been backed up (both the data and the configuration), it is recommended that you do not run the `amtune-directory` script in CHANGE mode to apply to the tuning changes. Review the tuning recommendations from REVIEW mode and apply the changes manually, if they meet your deployment needs.

Before making the tuning changes, the `amtune-directory` script stops and backs up Directory Server.

If you are working with a pilot or prototype Directory Server and you are sure you want to apply the tuning changes, follow these steps:

1. Back up both your Directory Server data and configuration.
2. Set the following parameter in the `amtune-directory` script:

   ```
   AMTUNE_MODE="CHANGE"
   ```
3. Run the `amtune-directory` script in CHANGE mode. For example:

   ```
   # ./amtune-directory dirmanager_password
   ```

   The *dirmanager_password* is the Directory Manager password.
4. Check the `amtune` log file for the results of the run.

   The script creates the log file in the same directory with the tuning scripts.

◆ ◆ ◆ CHAPTER 4

Distributed Authentication UI Server Tuning

If you have deployed a Distributed Authentication UI server, you can run the Access Manager tuning scripts to tune the Solaris or Linux operating system and the web container. Except for the `amtune-identity` and `amtune-prepareDSTuner` scripts, the tuning scripts do not require an instance of Access Manager server to run.

This chapter provides the following information:

- "Copying the Tuning Scripts" on page 41
- "Tuning the Operating System" on page 42
- "Tuning a Distributed Authentication UI Server Web Container" on page 42
- "Improving Performance for the Default Application User" on page 43

For more information about a Distributed Authentication UI server, see Chapter 11, "Deploying a Distributed Authentication UI Server," in *Sun Java System Access Manager 7.1 Postinstallation Guide*.

Copying the Tuning Scripts

Because Access Manager server is not installed on the system where the Distributed Authentication UI server is deployed, you must copy the following tuning scripts and files from an Access Manager 7.1 server installation:

- `amtune-os`, if you plan to tune the Solaris or Linux OS
- Appropriate web container script:
  - Web Server 7.0: `amtune-ws7`
  - Web Server 6.1 2005Q4: `amtune-ws61`
  - Application Server Enterprise Edition 8.2: `amtune-as8`
  - Application Server 7: `amtune-as7`
- `amtune-env` configuration file and `amtune-utils` script

The scripts and files are available on an Access Manager server installation in the following directory, depending on your platform:

- Solaris systems: *AccessManager-base*`/SUNWam/bin/amtune`
- Linux systems: *AccessManager-base*`/identity/bin/amtune`
- Windows systems: *javaes-install-directory*`\identity\bin\amtune`

AccessManager-base is the Access Manager 7.1 base installation directory. The default base installation directory is `/opt` on Solaris systems and `/opt/sun` on Linux systems.

On Windows systems, the default value for *javaes-install-directory* is `C:\Program Files\Sun\JavaES5`.

Tuning the Operating System

To tune the operating system for the Distributed Authentication UI server, run the `amtune-os` script. This script tunes the operating system kernel and TCP/IP parameters for both the Solaris OS and Linux OS. The script determines the OS type from the `uname -s` command.

On Solaris 10 and higher systems, the `amtune-os` script will not run if the wrapper `amtune` script is run in a local zone.

To run the `amtune-os` script, you first must copy it from an Access Manager server installation, as described in "Copying the Tuning Scripts" on page 41.

Tuning a Distributed Authentication UI Server Web Container

After you deploy the Distributed Authentication UI server on a web container, you can tune the web container by running the appropriate web container tuning script:

Web Container	Tuning Script
Web Server 7.0	`amtune-ws7`
Web Server 6.1	`amtune-ws61`
Application Server Enterprise Edition 8.2	`amtune-as8`
Application Server 7	`amtune–as7`

▼ To tune a Distributed Authentication UI server web container:

1. **Make sure you have copied the necessary scripts from an Access Manager server installation, as described in "Copying the Tuning Scripts" on page 41.**
2. **Edit the parameters in the `amtune-env` configuration file to specify the specific web container and tuning options.**
3. **To run the script in REVIEW mode, set `AMTUNE_MODE=REVIEW` in the `amtune-env` file.**
4. **Run the web container tuning script in REVIEW mode.**

   In REVIEW mode, the tuning script suggests tuning recommendations but does not make any changes to the deployment.
5. **Review the tuning recommendations in the output log file, which is available in the same directory as the tuning scripts.**

   If needed, make changes to the `amtune-env` file based on this run.
6. **To run the script in CHANGE mode, set `AMTUNE_MODE=CHANGE` in the `amtune-env` file.**
7. **To make actual tuning changes to your deployment, run the script in CHANGE mode.**
8. **Check the tuning results in the output log file.**

Improving Performance for the Default Application User

When you deploy a Distributed Authentication UI server using the default application user, performance can drop significantly due to the default application user's restricted privileges in Directory Server.

▼ To improve performance for the Distributed Authentication UI server default user:

1. **In the Access Manager console, create a new user. For example: `DistAuthUIuser`.**
2. **In Directory Server, add the `DistAuthUIuser` user with a new ACI to allow reading, searching, and comparing user attributes. An example of this new ACI is:**

```
dn:ou=1.0,ou=SunAMClientData,ou=ClientData,dc=example,dc=com
changetype:modifyadd:aci
```

```
aci: (target="ldap:///ou=1.0,ou=SunAMClientData,ou=ClientData,dc=example,dc=com")
(targetattr = "*"(version 3.0; acl "SunAM client data access for application user";
allow (read, search, compare)
userdn = "ldap:///uid=DistAuthUIuser,ou=people,dc=example,dc=com";)
```

3 On the Distributed Authentication UI server, set the following variables in the configuration file:

```
APPLICATION_USER=DistAuthUIuser
APPLICATION_PASSWD=DistAuthUIuser-password
```

On Solaris and Linux systems, the configuration file is based on the `amsamplesilent` file and is named `DistAuth_config` in the next step. Set any other variables in the `DistAuth_config` file, as required for your deployment.

On Windows systems, use the `AMConfigurator.properties` file to create a new configuration file. For example: `AMConfigurator-distauth.properties`.

4 Run the `amconfig` script using the edited configuration file.

For example, on a Solaris system with Access Manager installed in the default directory:

```
# cd /opt/SUNWam/bin
# ./amconfig -s ./DistAuth_config
```

On Windows systems, in the `amconfig.bat` file, change `AMConfigurator.properties` to `AMConfigurator-distauth.properties`, and then run the edited `amconfig.bat` file.

5 Restart the web container on the Distributed Authentication UI server.

◆ ◆ ◆ CHAPTER 5

Tuning Considerations

- "Operating System (OS) Considerations" on page 45
- "Third-Party Web Containers" on page 49

Note – The following tuning considerations are based on the tuning of various test deployments. Because each deployment is unique, you might need further customization and interactive testing to satisfy your specific requirements.

Operating System (OS) Considerations

Solaris OS

- "Sun Fire T1000 and T2000 Servers" on page 45
- "Solaris SPARC Systems with CMT Processor with CoolThreads Technology" on page 46

Sun Fire T1000 and T2000 Servers

If Access Manager is installed on a Sun Fire T1000 or T2000 server, the tuning scripts for Web Server 7.0 and Application Server 8.2 set the `JVM GC ParallelGCThreads` parameter to 8:

```
-XX:ParallelGCThreads=8
```

This parameter reduces the number of garbage collection threads, which could be unnecessarily high on a 32-thread capable system. However, you can increase the value to 16 or even 20 for a 32 virtual CPU machine such as a Sun Fire T1000 server, if it minimizes full garbage collection activities.

Solaris SPARC Systems with CMT Processor with CoolThreads Technology

For Solaris SPARC systems with CMT processor with CoolThreads technology, in the `/etc/opt/SUNWam/config/AMConfig.properties` file, it is recommended that you add the following properties at the end of the file:

```
com.sun.identity.log.resolveHostName=false
com.sun.am.concurrencyRate=value
```

where *value* depends on the number of cores in a Sun Fire T1000 or T2000 server. For example, for 8 cores, set *value* to 8, or for 6 cores, set *value* to 6.

Linux OS

To tune for maximum performance on Linux systems, you need to make tuning adjustments to the following items:

- "File Descriptors" on page 46
- "Virtual Memory" on page 48
- "Network Interface" on page 48
- "Disk I/O Settings" on page 48
- "TCP/IP Settings" on page 48

Note – If you are running Application Server 8.1 on Red Hat Linux, the stack size of the threads created by the Red Hat OS for Application Server is 10 Mbytes, which can cause JVM resource problems (CR 6223676). To prevent these problems, set the Red Hat OS operating stack size to a lesser value such as 2048 or even 256 Kbytes, by executing the `ulimit` command before you start Application Server. Execute the `ulimit` command on the same console that you will use to start Application Server. For example:

```
ulimit -s 256
```

File Descriptors

You might need to increase the number of file descriptors from the default. Having a higher number of file descriptors ensures that the server can open sockets under high load and not abort requests coming in from clients. Start by checking system limits for file descriptors with this command:

```
cat /proc/sys/fs/file-max
8192
```

The current limit shown is 8192. To increase it to 65535, use the following command (as `root`):

```
echo "65535" > /proc/sys/fs/file-max
```

To make this value to survive a system reboot, add it to `/etc/sysctl.conf` and specify the maximum number of open files permitted:

```
fs.file-max = 65535
```

Note: The parameter is not `proc.sys.fs.file-max`, as you might expect.

To list the available parameters that can be modified using `sysctl`:

```
sysctl -a
```

To load new values from the `sysctl.conf` file:

```
sysctl -p /etc/sysctl.conf
```

To check and modify limits per shell, use the following command:

```
limit
```

The output will look something like this:

```
cputime         unlimited
filesize        unlimit
datasize        unlimited
stacksize       8192 kbytes
coredumpsize    0 kbytes
memoryuse       unlimited
descriptors     1024
memorylocked    unlimited
maxproc         8146
openfiles       1024
```

The `openfiles` and `descriptors` show a limit of 1024. To increase the limit to 65535 for all users, edit `/etc/security/limits.conf` as `root`, and modify or add the `nofile` setting (number of file) entries:

```
*         soft    nofile                    65535
*         hard    nofile                    65535
```

The asterisk (*) is a wildcard that identifies all users. You could also specify a user ID instead.

Then edit `/etc/pam.d/login` and add the line:

```
session required /lib/security/pam_limits.so
```

On Red Hat Linux , you also need to edit `/etc/pam.d/sshd` and add the following line:

```
session required /lib/security/pam_limits.so
```

On many systems, this procedure will be sufficient. Log in as a regular user and try it before doing the remaining steps. The remaining steps might not be required, depending on how pluggable authentication modules (PAM) and secure shell (SSH) are configured.

Virtual Memory

To change virtual memory settings, add the following to `/etc/rc.local`:

```
echo 100 1200 128 512 15 5000 500 1884 2 > /proc/sys/vm/bdflush
```

For more information, view the man pages for `bdflush`.

Network Interface

To ensure that the network interface is operating in full duplex mode, add the following entry into `/etc/rc.local`:

```
mii-tool -F 100baseTx-FD eth0
```

where `eth0` is the name of the network interface card (NIC).

Disk I/O Settings

To tune disk I/O performance for a non-SCSI disk, follow these steps:

1. Test the disk speed with this command:

   ```
   /sbin/hdparm -t /dev/hdX
   ```

2. Enable direct memory access (DMA) with this command:

   ```
   /sbin/hdparm -d1 /dev/hdX
   ```

3. Check the speed again using the `hdparm` command. Given that DMA is not enabled by default, the transfer rate might have improved considerably. In order to do this at every reboot, add the `/sbin/hdparm -d1 /dev/hdX` line to `/etc/conf.d/local.start`, `/etc/init.d/rc.local`, or whatever the startup script is called.

TCP/IP Settings

To tune the TCP/IP settings, follow these steps:

1. Add the following entry to `/etc/rc.local`:

   ```
   echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
   echo 60000 > /proc/sys/net/ipv4/tcp_keepalive_time
   echo 15000 > /proc/sys/net/ipv4/tcp_keepalive_intvl
   echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
   ```

2. Add the following to `/etc/sysctl.conf`:

```
# Disables packet forwarding
net.ipv4.ip_forward = 0
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
# Disables the magic-sysrq key
kernel.sysrq = 0
net.ipv4.ip_local_port_range = 1204 65000
net.core.rmem_max = 262140
net.core.rmem_default = 262140
net.ipv4.tcp_rmem = 4096 131072 262140
net.ipv4.tcp_wmem = 4096 131072 262140
net.ipv4.tcp_sack = 0
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_keepalive_time = 60000
net.ipv4.tcp_keepalive_intvl = 15000
net.ipv4.tcp_fin_timeout = 30
```

3. Add the following as the last entry in `/etc/rc.local`:

```
sysctl -p /etc/sysctl.conf
```

4. Reboot the system.
5. Use this command to increase the size of the transmit buffer:

```
tcp_recv_hiwat ndd /dev/tcp 8129 32768
```

Third-Party Web Containers

- "IBM WebSphere Application Server" on page 49
- "BEA WebLogic Server" on page 51

IBM WebSphere Application Server

Consider making the following changes in the WebSphere Administrative Console:

- "JVM Tuning Parameters" on page 50
- "Servlet Caching" on page 50
- "Thread Pool Size" on page 51

For more information, see the "IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series" at:

http://www.redbooks.ibm.com/
/Redbooks.nsf/RedbookAbstracts/sg246198.html?OpenDocument

JVM Tuning Parameters

Add the JVM tuning parameters for JVM 1.4.2 shown below, by following these links in the console:

```
Servers>Application Servers>server1>Process Definition>Java Virtual Machine
```

Add "`-server`" as the first parameter in the "`Generic JVM arguments`" box. Then, add the following entries after the other existing parameters:

```
-XX:NewSize=336M -XX:MaxNewSize=336M
-XX:+DisableExplicitGC
-XX:+UseParNewGC
-XX:+UseConcMarkSweepGC
-XX:+CMSPermGenSweepingEnabled
-XX:+UseCMSCompactAtFullCollection
-XX:CMSFullGCsBeforeCompaction=0
-XX:+CMSClassUnloadingEnabled
-XX:SoftRefLRUPolicyMSPerMB=0
-XX:+PrintClassHistogram
-XX:+PrintGCTimeStamps
-Xloggc:/opt/WebSphere/AppServer/logs/server1/gc.log
-XX:-CMSParallelRemarkEnabled
```

If you use WebSphere 6.x with Sun JVM 1.5 or later, then some of the garbage collection (GC) algorithms can be safely removed. The following is a list of JVM options that can be used with Sun JVM 1.5 or later.

```
-XX:NewSize=336M -XX:MaxNewSize=336M
-XX:+DisableExplicitGC
-XX:+UseParNewGC
-XX:+UseConcMarkSweepGC
-XX:+PrintClassHistogram
-XX:+PrintGCTimeStamps
-Xloggc:/opt/WebSphere/AppServer/logs/server1/gc.log
-XX:-CMSParallelRemarkEnabled
```

Servlet Caching

Make sure that servlet caching is enabled by checking the checkbox next to "Enable servlet caching" by following these links in the console:

```
Application Servers>server1>Web Container>Configuration: Servlet caching
```

Thread Pool Size

Allow the thread pool to grow beyond the maximum thread pool size set by checking the checkbox next to "Allow thread allocation beyond maximum thread size" by following these links:

```
Application Servers>server1>Web Container>Thread Pool Is Growable
```

BEA WebLogic Server

Consider making the following changes:

- "JVM GC Parameter" on page 51
- "Heap Size" on page 52
- "Execute Queue Thread Count" on page 52
- "Connection Backlog Buffering" on page 53

JVM GC Parameter

For BEA WebLogic Server 8.1 SP4, to avoid the `java.lang.OutofMemoryError` reported by the WebLogic JVM 1.4.2_05, add the following JVM GC (garbage collection) parameter in the `startWebLogic.sh` `JAVA_OPTIONS`:

```
-XX:-CMSParallelRemarkEnabled
```

Set this parameter in addition to the other heap size and GC parameters that have been added for JVM 1.4.2 for Application Server 8.1 and Web Server 6.1.

For example, if Access Manager is installed in the default `user_projects` location (`/usr/local/bea/user_projects/domains/mydomain/startWebLogic.sh`):

```
JAVA_OPTIONS="-XX:+DisableExplicitGC -XX:+UseParNewGC
-XX:+UseConcMarkSweepGC -XX:+CMSPermGenSweepingEnabled
-XX:+UseCMSCompactAtFullCollection -XX:CMSFullGCsBeforeCompaction=0
-XX:+CMSClassUnloadingEnabled -XX:-CMSParallelRemarkEnabled
-XX:SoftRefLRUPolicyMSPerMB=0 -XX:+PrintClassHistogram
-XX:+PrintGCTimeStamps
-Xloggc:/usr/local/bea/user_projects/domains/mydomain/myserver/gc.log"
```

If you use WebLogic 9.x with Sun JVM 1.5 or later, then some of the GC algorithms can be safely removed. The following is a list of JVM options that can be used with Sun JVM 1.5 or later.

```
-XX:NewSize=336M -XX:MaxNewSize=336M
-XX:+DisableExplicitGC
-XX:+UseParNewGC
```

```
-XX:+UseConcMarkSweepGC
-XX:+PrintClassHistogram
-XX:+PrintGCTimeStamps
-Xloggc:/opt/WebSphere/AppServer/logs/server1/gc.log
-XX:-CMSParallelRemarkEnabled
```

Heap Size

Modify the `commonEnv.sh` script in the `/usr/local/bea/weblogic81/common/bin` directory for heap size increases in the section where `$PRODUCTION_MODE" = "true"` (which should be set to true, before running Access Manager in `/usr/local/bea/user_projects/domains/mydomain/startWebLogic.sh`):

```
# Set up JVM options base on value of JAVA_VENDOR
if [ "$PRODUCTION_MODE" = "true" ]; then
  case $JAVA_VENDOR in
  BEA)
    JAVA_VM=-jrockit
    MEM_ARGS="-Xms128m -Xmx256m"
  ;;
  HP)
    JAVA_VM=-server
    MEM_ARGS="-Xms32m -Xmx200m -XX:MaxPermSize=128m"
  ;;
  IBM)
    JAVA_VM=
    MEM_ARGS="-Xms32m -Xmx200m"
  ;;
  Sun)
    JAVA_VM=-server
    MEM_ARGS="-Xms2688M -Xmx2688M -XX:NewSize=336M -XX:MaxNewSize=336M"
   # MEM_ARGS="-Xms32m -Xmx200m -XX:MaxPermSize=128m"
```

Execute Queue Thread Count

Set the Execute Queue Thread count to be more than the number of CPUs. For example, consider using a value that is twice the number of CPUs. Set this value in either the console or in the `/usr/local/bea/user_projects/domains/mydomain/config.xml` file:

```
<ExecuteQueueName="MyExecute Queue" ThreadCount="8" ThreadsIncrease="4"/>
```

For more information, see “Modifying the Default Thread Count” in “WebLogic Server Performance and Tuning” at:

`http://e-docs.bea.com/wls/docs81/perform/WLSTuning.html#1142218`

Connection Backlog Buffering

A guideline for setting Connection Backlog Buffering is 8192 for a server with 4 Gbytes of physical memory (which is equivalent to the `ConnectionQueue` size tuning set in the Sun Java System Web Server 6.1 `magnus.conf` file).

For more information, see "Tuning Connection Backlog Buffering" in the "WebLogic Server Performance and Tuning" document at:

`http://e-docs.bea.com/wls/docs81/perform/WLSTuning.html#1136287`

PART II

Troubleshooting Performance Issues

- Chapter 6, "Best Practices for Performance Tuning and Testing,"
- Chapter 7, "Advanced Performance Tuning,"

◆ ◆ ◆ CHAPTER 6

Best Practices for Performance Tuning and Testing

Using a planned, systematic approach to tuning will help you avoid most performance troubleshooting pitfalls. This chapter includes the following topics:

- "Avoiding Common Performance Testing and Tuning Mistakes" on page 57
- "Using a Systematic Approach to Performance Tuning" on page 58

Avoiding Common Performance Testing and Tuning Mistakes

Don't make the same mistakes deployment engineers and performance test teams usually make. Deployment engineers usually construct the system and perform the functional tests. Next the engineers hand over the system to the performance testing team. The testing team develops test plans and test scripts based on the targeted load assumptions. The project manager usually gives the testing team only a few hours or a few days to conduct the performance tests. Using this approach, the testing team usually encounters unexpected behaviors during the tests.

The testing team then realizes that performance tuning was not done been before the tests. Tuning is hastily done, but problems still persist. The testing team starts to experiment with different parameter settings and configurations. This frequently leads to more problems which jeopardize the schedule. Even when the testing team successfully produces a performance report, the report usually fails to cover test cases and information crucial to capacity planning and production launch support. For example, the report often does not capture the system capacity, request breakdowns, and the system stability under stress.

You can avoid these performance testing and tuning mistakes by using a systematic approach, and by allocating adequate project resources and time.

Using a Systematic Approach to Performance Tuning

The best practice is a systematic approach to performance testing with an allocation of a minimum of three weeks testing time. A good performance tuning plan includes the following phases:

1. "Constructing the System" on page 58
2. "Automated Performance Tuning" on page 58
3. "Related Systems Tuning" on page 58
4. "Baseline Modular Performance Testing" on page 59
5. "Advanced Performance Tuning" on page 67
6. "Targeted Performance Testing" on page 67

Constructing the System

During the system construction phase, the entire system is built step by step in a modular fashion. For a detailed example, see the document *Deployment Example 1: Access Manager 7.1 Load Balancing, Distributed Authentication UI, and Session Failover*. Each module is in the example is built and then verified. It's always easier to verify a module build than to troubleshoot an entire system. The modular verification tests prevent configuration problems from being buried in the system. Some of these verification steps are performance related. For example, there are steps to verify that sticky load balancing is working properly. See "To Configure the Access Manager Load Balancer" in *Deployment Example 1: Access Manager 7.1 Load Balancing, Distributed Authentication UI, and Session Failover*

Automated Performance Tuning

In this phase, you tune the system using the automated tuning script `amtune` that comes with the product. The `amtune` script automates most of the performance tunings and address most, if not all, Access Manager tuning needs. Manual tweaking is unnecessary and may cause harm unless you run into some of the known extreme problems

Related Systems Tuning

In this phase, you manually tune Directory Server, any Web Servers that host Web Policy Agents, and any Application Servers that host J2EE Policy Agents. The typical tuning steps are as follows:

1. Run `amtune` to tune the Access Manager system. For more detailed information, see Chapter 2, "Access Manager Tuning Scripts."
2. Follow the `amtune` onscreen prompts to tune the related Directory Server configuration instances. The following is an overview of the primary tuning steps you must complete:

a. Increase the `nsslapd-dbcachesize` value.
b. Relocate `nsslapd-db-home-directory` to the `/tmp` directory.

For detailed information, see the Directory Server documentation.

3. Manually tune the user Directory Server user database instance if one is used. The following is an overview of the primary tuning steps you must complete:

   a. Increase the `nsslapd-dbcachesize` value.
   b. Relocate the `nsslapd-db-home-directory`to the `/tmp` directory.

4. If the Access Manager sub-realm is pointing to an external Directory Server user database, then manually tune the sub-realm LDAP connection pool.

   The `amtune` script tunes only the LDAP connection pools of the root realm. See "Tuning the LDAP Connection Pool and LDAP Configurations" on page 69. You can configure the following parameters on `LDAPv3IDRpo`:

   a. LDAP Connection Pool Minimum Size
   b. LDAP Connection Pool Maximum Size

5. If you have installed a Web Policy Agent on a Sun Web Server, then manually tune the Web Server. You must configure the following parameters in the `magnus.conf`:

   - `RqThrottle`
   - `RqThrottleMin`
   - `RqThrottleIncrement`
   - `ConnQueueSize`

   If Access Manager is deployed on a Sun Web Server, the `amtune` script will modify the Web Server `magnus.conf` file. You can copy the changes and use the changed values in the Web Policy Agent Web Server.

6. If you have installed a J2EE Policy Agent on an application server, see"Third-Party Web Containers" on page 49 for instructions on manually tune both the J2EE Policy Agent and the application server. You must configure settings for heap sizes and for garbage collection (GC) behaviors.

Baseline Modular Performance Testing

The system is largely performance tuned after you've run the `amtune` script. But it is still too early to perform the final complex performance tests. It's always more difficult to troubleshoot performance problems in the entire system than to troubleshoot individual system components performing basic transactions. So in this phase, you perform several baseline tests. Be sure that the specific baseline test scripts you write will:

- Verify the functions of the sub-systems under the stress load of basic transactions such as authentications and authorizations.
- Establish baseline performance benchmarks for basic transactions.

Conducting Baseline Authentication Tests

You will need the following test scripts are to generate the basic authentication workload:

- Login and logout test
- Login and time out test

For all tests, randomly pick user IDs from a large user pool, from minimally 100K to one million users. The load test script should first log the user in, then either log the user out or simply drop the session and let the session time out. A good practice is to remove all static pages and graphics requests from the scripts. This will make the workload cleaner and well— defined. The results are easier to interpret.

The test scripts should have zero think time to put the maximum workload on the system. The tests are not focused on response times in this phase. The baseline tests should determine the maximum system capacity based on maximum throughput. The number of test users, sometimes called test threads, is usually a few hundred. The exact number is unimportant. What is important is to achieve as close to 100% Access Manager CPU usage as possible while keeping the average response time to at least 500 ms. A minimum of 500 ms is used to minimize the impact of relatively small network latencies. If the average response time is too low (for example 50ms), a large portion is likely to be caused by network latency. The data will be contaminated with unnecessary noise.

Determine the Number of Test Users

In the following example baseline test, 200 users per one AM instance are used. For your tests, you could use 200 users for one Access Manager instance, 400 users for two Access Manager instances, 600 users for three Access Manager instances, and so forth. If the workload is too low, start with 100 users, and increase it by increments of 100 to find out the minimum number. Once you have determined the minimum test users per AM instance, use with this number for the rest of the tests to make the results more comparable.

Determine the System Steady State

In the example baseline tests, the performance data is captured at the steady state. The system can take any where from 5 to 15 minutes to reach its steady state. Watch the tests. The following indicators will settle into predictable patterns when the system has reached its steady state:

- Transactions per second (TPS), also called throughput
- Average response time of individual transactions
- CPU usage of all affected servers (including Access Manager, Directory Server, and any load generation machines)
- Number of transactions performed by each component in a given period, categorized by transaction types (see Appendix for details)

The following are examples of capturing transactions by categories on different sytems.

On each Access Manager host, parse the container access log to gather the number of different transactions received. For example, if Access Manager is deployed on Sun Web Server, use the following command to obtain the result:

```
cd /opt/SUNwbsvr/https-<am1host>/logs
cp access a; grep Login a | wc; grep naming a | wc; grep session a|
wc; grep policy a | wc ; grep jaxrpc a | wc; grep notifi a | wc;
grep Logout a | wc; wc a;
```

On each LDAP server, parse the LDAP access log to gather the number of different transactions received. For example, use the following command to obtain the result:

```
cd <slapd-xxx>/logs
cp access a; grep BIND a | grep "uid=u" | wc; grep BIND a|wc;
grep UNBIND a| wc; grep SRCH a| wc; grep RESULT a| wc; wc a ;
```

Conduct the Baseline Test

In this example, the baseline test follows this sequence:

1. Log in and log out on each individual AM directly.
2. Log in and time out on each individual AM directly.
3. Log in and log out using a load balancer with one Access Manager server.
4. Log in and time out using a load balancer with one Access Manager server.
5. Log in and log out test on LB with two AM instances behind
6. Perform login and timeout test on LB with two AM instances behind

If you have two Access Manager instances behind a load balancer, the above tests actually involve at least ten individual test runs: two test runs for 1 through 4, one test run, and one test run for 6.

Note – In order to perform any log in and timeout test, you must reduce the maximum session timeout value to lower than the default value. For example, change the default 30 minutes to one minute. Otherwise, at the maximum throughput, there will be too many sessions lingering on the system for so long that the memory will be exhausted quickly.

Analyze the Baseline Test Results

The data you capture will help you identify possible trouble spots in the system. The following are examples of things to look for in the baseline test results.

Compare the maximum authentication throughput of individual Access Manager instances with no load balancer in place.

If identical hardware is used in the test, the number of authentication transactions per second should be roughly the same for each Access Manager instance. If there is a large variance in throughput, investigate why one server behaves differently than another.

Compare the maximum authentication throughput of individual Access Manager instances that have a load balancer in front of them.

Using a load balancer should not cause a decrease in the maximum throughput. In the example above, test 3 should yield results similar to test 1 results, and test 4 should yield results similar to test 2 results. If the maximum throughput numbers go down when a load balancer is added to the system, investigate why the load balancer introduces significant overhead. For example, you could conduct a further test with static pages through the load balancer.

Verify that the maximum throughput on a load balancer with two Access Manager instances is roughly twice the throughput on a load balancer with one Access Manager instance behind it.

If the throughput numbers to do not increase proportionately with the number of Access Manager instances, you have not configured sticky load balancing properly. Users logged in to one Access Manager instance are being redirected to another instance for logout. You must correct the load balancer configuration. For related information, see "Configuring the Access Manager Load Balancer" in *Deployment Example 1: Access Manager 7.1 Load Balancing, Distributed Authentication UI, and Session Failover*.

Verify that for each test, the Access Manager transaction counts report indicates no unexpected Access Manager requests.

For example, if you perform the Access Manager login and logout test, your test results may look similar to this:

```
1581   15810  139128
   0       0       0
   0       0       0
   0       0       0
   0       0       0
   0       0       0
1609   16090  146419
3198   31972  286043 a
```

This output indicates three important pieces of information. First, the system processed 1581 login requests and 1609 logouts request. They are roughly equal. This is expected as each login is followed by one logout. Secondly, all other types of AM requests were absent. This is expected. Lastly, the total number of requests received, 3198, is roughly the sum of 1581 and 1609. This indicates there are no unexpected requests that we didn't `grep`in the command.

Troubleshoot the Problems You Find

A common problem is that when two Access Manager instances are both running, you see not only login and logout requests, but session requests as well. The test results may look similar to this:

```
  3159   31590  277992
     0       0       0
  5096   50960  486676
     0       0       0
     0       0       0
  1305   13050  127890
  3085   30850  280735
 12664  126621 1174471 a
```

In this example, for each logout request, there are now extra session and notification requests. The total number of requests does add up. This means there are no other unexpected requests. The reason for the session request is that the sticky load balancing is not working properly. A user logged in on one Access Manager instance, then is sent to another AM instance for logout. The second Access Manager instance must generate an extra session request to the originating AM instance to perform the request. The extra session request increases the system workload and reduces the maximum throughput the system can provide. In this case, the two Access Manager instances cannot double the throughout of the single Access Manager throughput. Instead, there is a mere 20% increase. You can address the problem at this point by reconfiguring the load balancer. This is an example of a problem should have been caught during modular verification steps in the system construction phase.

Run Extended Tests for System Stability

Once the system has passed all the basic authentication tests, it's a good practice to put the system under the test workload for an extended period of time to test the stability. You can use test 6 let it run over several hours. You may need to set up automated scripts to periodically remove excessive access logs generated so that they do not fill up the file systems.

Conducting Baseline Authorization Tests

You will need the following test scripts are to generate the basic authorization workload:

- Login, access an agent-protected page twice, logout test.

In this example, the baseline authorization test follows this sequence:

- Perform login, page-access and logout test on each individual Access Manager instance, with no load balancer in place.

  This test determines the Access Manager capacity without the influence of a network element such as the load balancer.

- Perform login, page-access and logout test on the load balancer with only one Access Manager instance behind it.

  This test determines the impact of the load balancer.
- Perform login, page-access and logout test on the load balancer with two Access Manager instances behind it.

  This test determines the baseline results when multiple Access Manager instances are running, and indicate whether the sticky load balancing is configured properly.

It is a good practice to set up a single URL policy that allows all authenticated users to access the wildcard URL protected by the policy agent. This simplified setup keep things simple in the baseline tests.

For all tests, randomly pick user IDs from a large user pool, from minimally 100K to one million users. The load test scripts log the user in, accesses a protected static page twice, and then logs the user out. A good practice is to remove all other static page or gif requests from the scripts. This will make the workload cleaner, well-defined, and the results are easier to interpret.

The test scripts should have zero think time to put the maximum workload on the system. The tests are not focused on response times in this phase. The baseline tests should determine the maximum system capacity based on maximum throughput. The number of test users, sometimes called test threads, is usually a few hundred. The exact number is unimportant. What is important is to achieve as close to 100% Access Manager CPU usage as possible while keeping the average response time to at least 500 milliseconds. A well executed test indicates the maximum system capacity while minimizing the impact of network latencies.

Determine the Number of Test Users

A typical 200 users per one Access Manager instance can be used . For example, you could use 200 users for one Access Manager instance, 400 users for two Access Manager instances, 600 users for three Access Manager instances, and so on. If the workload is too low, start with 100 users, and increase it by a 100—user increments to find out the minimum number. Once the number of test users per Access Manager instance is determined, continue to use this number for the rest of the tests to make the results more comparable. If you have two Access Manager instances behind a load balancer, the above tests actually involve at least five individual test runs. You conduct two runs each for tests 1 and 2, and conduct one run for test 3.

Verify that for each test, the response time of the second protected resource access is significantly lower than the response time of the first protected page access. On the first access to a protected resource, the agent needs to perform uncached session validation and authorization. This involves the agent communicating with Access Manager servers. On the second access to a protected resource, the agent can perform cached session validation and authorization. The agent does not need to communicate with the Access Manager servers. Thus the second access tends to be significantly faster. It's common to see the first page access takes 1 second (this highly depends on the number of test users used), while the second page access

takes less than 10 ms (this does not depend too much on the number of test users used). If the second page access is not as fast as it should be, compared with the first page access, you should investigate to find out why. Is it because first page access being relatively too fast ? If so, you can increase the number of test users to increase the response time of the first page access. Is it because the agent machine is undersized so that no matter how much load you put on the system, Access Manager does not reach full capacity, and the agent machine reaches full capactiy first. In this case, since the agent machine is the bottleneck, and not the AccessManager, you can expect both the first and second page access to be slow while Access Manager responds quickly.

Analyze the Test Results

The data you capture will help you identify possible trouble spots in the system. The following are examples of things to look for in the baseline test results.

Compare the maximum authorization throughput of individual Access Manager instances with no load balancer in place.

If identical hardware is used in the test, the number of authorizations transactions per second should be roughly the same for each Access Manager instance. If there is a large variance in throughput, investigate why one server behaves differently than another.

Compare the maximum authorization throughput of individual Access Manager instances that have a load balancer in front of them.

Using a load balancer should not cause a decrease in the maximum throughput. In the example above, test 2 should yield results similar to test 1 results. If the maximum throughput numbers go down when a load balancer is added to the system, investigate why the load balancer introduces significant overhead. For example, you could conduct a further test with static pages through the load balancer.

Verify that the maximum throughput on a load balancer with two Access Manager instances is roughly twice the throughput on a load balancer with one Access Manager instance behind it.

If the throughput numbers to do not increase proportionately with the number of Access Manager instances, you have not configured sticky load balancing properly. Users logged in to one Access Manager instance are being redirected to another instance for logout. You must correct the load balancer configuration. When sticky load balancing is properly configured, each Access Manager should serve requests independently and thus the system would scale near linearly. If the throughput numbers to do not increase proportionately with the number of Access Manager instances, you have not configured sticky load balancing correctly. For related information, see “Configuring the Access Manager Load Balancer” in *Deployment Example 1: Access Manager 7.1 Load Balancing, Distributed Authentication UI, and Session Failover*.

Verify that for each test, the Access Manager transaction counts report indicates no unexpected Access Manager requests.

For example, if you perform the Access Manager login and logout test, your test results should look similar to this:

```
1079   10790   94952
1032   10320   99072
1044   10440  101268
1064   10640  101080
   0       0       0
   0       0       0
1066   10660   97006
5312   53093  495052 a
```

This output indicates three pieces of information. First, the system processed 1079 login, 1032 naming, 1044 session, 1064 policy and 1066 logout requests. These numbers are roughly equal. For each login, there is one naming call, one session call (to validate the user's session), one policy call (to authorize the user's access) and one logout. Secondly, all other types of Access Manager requests were absent. This is expected. Lastly, the total number of request received 5312 is roughly the sum of login, naming, session, policy and logout requests. This indicates there are no unexpected requests that we didn't `grep` in the command.

Troubleshoot Problems You Find

A common problem is that when two AM instances are both running, you see the number of session requests exceeds the number of logins. For example, the test output may look similar to this:

```
 4075   40750  358600
 4167   41670  400032
19945  199450 1913866
 3979   39790  381984
    0       0       0
 3033   30330  297234
 3946   39460  359086
39194  391891 3713840 a
```

Note that for each login request, there are now 5 session requests, and 0.75 notifications. The total number of requests do add up though. This indicates there are no other unexpected requests. There more session requests per login because the sticky load balancing is not working properly. A user logged in on one Access Manager instance is sometimes sent to another Access Manager instance for session validation and logout. The second Access Manager instance must generate extra session and notification requests to the originating Access Manager instance to perform the request. The extra requests increase the system workload and reduce the maximum throughput the system can provide. In this case, the two Access Manager instances cannot double the throughout of the single AM throughput. You can address the problem by reconfiguring the load balancer. The problem should have been caught during modular verification steps in the system construction phase.

Conduct Extended Stability Tests

Once you've passed all the basic authorization tests, it's a good idea to put the system under the workload for extended period of time to test the stability. You can use test 3 and let it run over several hours. You may need to set up automated scripts to periodically remove excessive access logs generated so that they do not fill up the file systems.

Advanced Performance Tuning

The `amtune` script is specifically designed to address most, if not all, of the performance tuning needs. This means that you almost never need to manually tweak performance parameters. With the large number of performance related parameters, tweaking them invite more problems instead of solving them. However, there are a few special situations that `amtune` currently does not tune or tune well. This is documented in Chapter 7, "Advanced Performance Tuning." For each special situation, there is an explanation of what `amtune` is doing today, how to identify whether you need to manually tune the parameters, and how to tune them. It is worth repeating here that most, if not all, of your performance tuning should be addressed by the `amtune` script. Performance problems are usually caused by poor system configuration. The special tuning cases should be used only if they actually apply to your specific case.

Targeted Performance Testing

By the time you've reached this test phase, you've already done enough baseline tests that give you both the confidence the system performs properly, and a rough idea of how the system should perform in your targeted performance test scenarios. Target performance tests typically have the projected real-world workload in mind. They usually include many more test users, but also slower users (by introducing realistic think time). The test also tries not to test the system at maximum CPU usage. Instead, the tests usually focus on several scenarios. Examples:

- Average workload that gauge the users' experience in terms of the average response time.
- Peak workload when demands peak or one or more servers are down, and load transfer has occurred, to gauge the users perceived average response time, and the system stability.
- Stability tests that use average or peak workload to run extended period of time, such as a day or a week.

Regardless what scenarios you are testing, if a problem occurs, it always helps to go back to the baseline tests to validate if certain things have changed in the environment, and to isolate the new elements (hardware or software configuration changes) that may have contributed to the problem. Unless you've isolated the problem, haphazardly tweaking performance related parameters is not productive, and usually do more harm and cause more confusion. Detailed troubleshooting methodology and techniques are beyond the scope of this document. See section name for suggestions on troubleshooting some common performance problems.

◆ ◆ ◆ CHAPTER 7

Advanced Performance Tuning

After conducting basic performance tuning and following the best practices recommendations described in previous chapters, you may still encounter performance issues. This chapter helps you troubleshoot the most common Access Manager performance issue. Topics in this chapter include:

- "Tuning the LDAP Connection Pool and LDAP Configurations" on page 69
- "Resolving Memory Issues" on page 71
- "Performance Tuning Questions and Answers" on page 74
- "More Resources" on page 77

Tuning the LDAP Connection Pool and LDAP Configurations

The `amtune` script provided by AccessManager recommends parameter values for the following three LDAP connection pools:

- Root Realm User Authentication LDAP Connection Pool
- Root Realm Data Store LDAP Connection Pool
- Access Manager Configuration Store and SMS LDAP Connection Pools

But the script does not actually tune the LDAP connection pools for you. You need to make the changes manually. In addition, in deployments with a subrealm, you must also manually tune the subrealm's connection pools. Just like the root realm, each sub-realm can have its own user authentication LDAP connection pool and data store LDAP connection pool. You must manually tune these as well.

You can modify one or more of the three LDAP connection pool configurations . In each configuration, the recommended values are MIN=8 and MAX=32. Under some conditions, you can increase the MAX value up to 64. The following sections describe how to manually tune the connection pools:

- "To Tune the User Authentication LDAP Configuration" on page 70
- "To Tune the Data Store LDAP Configuration" on page 70

- "To Tune the Access Manager Configuration Store and SMS LDAP Configuration" on page 70

To Tune the User Authentication LDAP Configuration

You can modify the settings on one of the following depending upon the module you use for user authentication.

LDAP Authentication Module
This module is used only to authenticate the user. In the Access Manager console, under Configuration, click Authentication > Core.

Data Store Authentication Module
When the Data Store is as the authentication module, the Data Store LDAP connection pool settings are used. No additional Authentication connection pool settings are used.

To Tune the Data Store LDAP Configuration

The Data Store LDAP Configuration is used for retrieving user profiles and can also be used for authentication. By default, Access Manager 7.1 supports two types of Data Store plug-ins: AMSDK and LDAPv3. If the Data Store Authentication module is used for authentication (see above), then the recommended Data Store LDAP configuration settings are MIN=8 and MAX=64. You can modify the settings on one of the following depending upon the Data Store plug-in you use:

AMSDK Configuration
The AMSDK LDAP configuration is stored in the `serverconfig.xml` file under the Access Manager `config` directory. The server group name is `default`.

LDAPv3 Configuration
To modify the LDAPv3 Configuration, in the Access Manager console, under Access Control, click Realm > DataStore.

To Tune the Access Manager Configuration Store and SMS LDAP Configuration

The Service Management (SMS) LDAP Configuration is used for storing and retrieving all Access Manager configuration and Policy Service configuration. The SMS LDAP Configuration is stored in the `serverconfig.xml` file under the Access Manager `config` directory. The server group is `sms`.

1. Start by setting all the connection pool configurations with MIN=8 and MAX=32.
2. If you must make adjustments based on performance test results, adhere to the following requirements:

- The MIN value should be at least 8.
- The MAX value for any pool should not be greater than 64. The MAX value of 32 is enough for most typical deployments.

Special requirements are outside the scope of this document.

3. After following steps 1 and 2, if low throughput or low response times persist, then try the following solutions:
   - Verify that the Directory Server instance is not at 100% CPU usage. If the Directory Server instance is at 100% and the throughput is still low, revisit the indexing on the Directory Server entries. Be sure that Directory Server indexing is configured properly.
   - Run load tests to verify that Access Manager login is not causing performance to slow down. First run the tests with logging enabled, and then run the tests with logging disabled. If you find that logging is causing low response time, then you can tune the logging service through the Access Manager console. See "Logging" in *Sun Java System Access Manager 7.1 Administration Reference*.

Resolving Memory Issues

The `amtune` script automatically tunes all memory related parameters. In most deployments, this is sufficient. However, occasionally the `amtune` tuning may not be sufficient and you may run into memory issues. Memory issues manifest themselves through excessively frequent garbage collection (GC) operations or frequent "Out of Memory" errors.

To resolve memory related issues, tune the following parameters:

- `com.iplanet.am.sdk.cache.maxSize`

  User cache/SDK cache.

- `com.iplanet.am.session.maxSessions`

  Max Active Session the system should allow.

- `com.iplanet.am.notification.threadpool.size`

  Number of threads to process session notifications.

- `com.iplanet.am.notification.threadpool.threshold`

  Notification Queue size.

- `com.iplanet.am.session.purgedelay`

  Number of minutes to delay the purge timed-out session.

All the parameters listed above can be tuned by editing the `AMconfig.properties` file which is located in under `/etc/opt/SUNWam/config` if installed using the JES installer. If the Access Manger is installed using the single WAR, than `AMConfig.properties` is located in directory you specified when you configured the WAR file.

The minimum required JVM heap size for Access Manager is 1024 mb.

Tuning `com.iplanet.am.session.maxSessions`
The tuning of this property entire depnds on the JVM Heap size configured in the web container where the Access Manager is deployed. The minimum required JVM heap size for Access Manager is 1024 mb and the # of sessions supported for 1024mb is 12000 and every additional 512mb can support upto 18000 sessions.

Tuning `com.iplanet.am.sdk.cache.maxSize`
The sdk cache size should be same as the value set for `com.iplanet.am.session.maxSessions`.

Tuning `com.iplanet.am.notification.threadpool.threshold`
This is the Notification Queue size. The Notification Queue size should be less than or equal to 30% of the Max Sessions.

The following table lists sample settings for the parameters listed above based on the rules described above.

Maximum JVM Heap Size	Maximum Active Sessions	SDK Cache Size	Notification Queue Size
1024mb	12000	12000	4000
1536mb	30000	30000	10000
2048mb	48000	48000	16000
2560mb	66000	66000	22000
3136mb	90000	90000	30000

The above settings may not be suitable for certain deployments. When the number of user attributes retrieved is large, the SDK cache size will increase. Similarly, if the Extra Session properties are set, the Session size will increase.

In these cases, use one of the following options to solve the memory related issues:

- Reduce the Max Sessions limit and make sure you follow the above rules. If you reduce the Max Sessions you may need to add additional instances to support additional sessions. If you do not want to add additional instances you can use the 64bit JVM.
- Reduce the SDK cache size. If you reduce the SDK cache size, your performance will go down. For better performance it is always better to set the SDK cache size equal to Max Sessions, and add additional instances to support more sessions.

To Tune the Notification Threadpool Size

Set the value of `com.iplanet.am.notification.threadpool.size` based on number of CPUs and based on the `purgedelay` value. See "To Tune the Purge Delay Settings" on page 73for related information.

- If `purgedelay` is set to 0, the threadpool should be set using the following formula: (number of CPUs) x 3 = threadpool size. For example, for a machine with 8 CPUs, the threadpool size is 24. ForNiagra T1000/T2000 machines, use the formula: (number of cores) x 3 = threadpool size.
- If the `purgedelay` value is set to greater than 0, then the threadpool should be set using the following formula: (number of CPUs) x 4 = threadpool size . ForNiagra T1000/T2000 machines, use the formula: (number of cores) x 4 = threadpool size. The `amtune` script currently does not set this value based on the above rules. This configuration should be done manually. With the above setting if you still see problems such as frequent "Cannont send notification" or "Notification task queue full" errors in the `amSession` debug file, this indicate that the `SessionNotification`queue is full. The problem could be related to the Policy Agent or SDK client which is receiving notifications. The Policy Agent or SDK client is not able to process notifications properly. Consider disabling notification mode on the Policy Agent.

To Tune the Purge Delay Settings

The `purgedelay` property is used to keep the session in memory in a timed-out state after the session has timed out. If the value is set to 0, then the session is removed from memory immediately. If the value is greater than zero, then the session is maintained in the memory until the `purgedelay` time elapses.

- In almost all deployments, `purgedelay` should be set to 0. The `amtune` script will set the value to 0 when run.
- In special cases when the `purgedelay` value is greater than 0, reduce the number of active sessions (`com.iplanet.am.session.maxSessions`). Additionally, increase the notification threadpool size (`com.iplanet.am.notification.threadpool.size`)

The property `com.iplanet.am.session.maxSessions` describes the maximum number of active sessions that the system will allow. When the `purgedelay` is set to 0, the total number of sessions (active sessions and timed-out sessions) in memory will be equal to the value set for `com.iplanet.am.session.maxSessions`. If `purgedelay` is greater than 0, then the total number of sessions (active and timed-out sessions) in memory can be greater than active sessions. The difference will be based on three factors: the `purgedelay` time , the percentage of timed-out sessions, and the authentication rate. Therefore, when `purgedelay` is greater than zero, the maximum active sessions value should be reduced accordingly.

The simple way to do this is to look in the AccessManager 7.1sp1 session stats file. The `amMasterSessionTable` shows the current and peak values for `maxSessions` (active sessions + timed-out sessions) and `maxActive` (only active sessions) sessions in memory . Based on this information, the `maxSessions` value in the stats file limit should not exceed the 90000 limit for a JVM heap size of 3136mb. When the `purgedelay` is set to 0, only one notification is sent when a session is removed from memory. When the `purgedelay` is greater than 0, then there will be two notifications for each timed-out session. The number of notifications for timed-out sessions are increased, and now more notification threads are needed. So the notification thread pool size should also be increased.

Performance Tuning Questions and Answers

Question: How can I improve authentication performance against any LDAP v3 data repository?

Answer: If the Profile Ignored option is selected in the Access Manager console (go to Realm > Authentication >Advanced Properties), performance may improve. However, improved performance is not guaranteed because the Profile Ignored option prevents applications and policy agents from retrieving the user's profile attributes. The `amtune` script automatically tunes the LDAP connection pool for the Access Manager root realm which points to the configuration Directory Server instance. But the `amtune` script will not tune the subrealm you created, the subrealm where the LDAP v3 data repository is configured. You may need to manually tune the LDAP connection pool. After tuning the LDAP connection pool, if poor performance persists, troubleshoot the LDAP v3 repository itself.

Consider limiting the time you spend troubleshooting authentication performance issues. Authentication usually contributes only a small portion of the overall system overhead. Authorizations tend to be slower and a lot more frequent than other processes. But each user session involves only one authentication and multiple authorizations.

Question: How do I set the JVM heap sizes and other JVM option tuning parameters for a Distributed Authentication UI web application?

Answer: The web container that will load the Access Manager Distributed Authentication UI web application should have the same heap sizes and the same JVM tuning settings as the web container that runs Access Manager. You can use `amtune-ws7`, `amtune-ws61` or `amtune-as8` which come with Access Manager 7.1. You don't need much CPU usage as for Access Manager server machines. It is hard to tell by what ratio one can reduce the number of CPU usage on a Distributed Authentication UI machine. The ratio can be 1:4 or less. Run some load tests for a specific scenario to determine a good ratio.

The reason why a Distributed Authentication UI web container needs the same JVM heap sizes and garbage collection tuning parameters as those for the Access Manager server web containers is that `amclientsdk` maintains the same number of Access Manager sessions on the client side as on the Access Manager server itself. A Request for Enhancement (CR 6465831) has been filed for removal of the Access Manager sessions in Distributed Authentication UI `amclientsdk` deployments.

Question: What is the impact of checking `notenforced_list` for a set of URIs or URLs on J2EE policy agents?

Answer: The performance impact of checking `notenforced_list` is negligible. In general, having a `notenforced_list` of commonly requested and static content improves the overall system performance.

Question: What is the impact of using the SSL , for example the NSS library version that comes with JES 4 installer, on the performance of Access Manager 7.0 deployed on Niagara boxes such as T1000 or T2000?

Answer: If Access Manager 7.0 was installed using JES 4 installer and its default SSL libraries, then the markedly improved performance that comes with NSS 3.11 may not be present and won't be used. Use the NSS libraries version 3.11 or higher when Access Manager 7.0 is deployed on Niagra T1000 or T2000 systems. Go to the Sunsolve web site for downloading the NSS libraries. Note that starting with JES 5 and Access Manager 7.1, the NSS libraries have been upgraded to a version higher than 3.11.

Question: Why is it so slow to create or delete users if I use a program based on `amsdk`, but much faster if I use the `ldapmodify` command?

Answer: If the same policy is modified for each user, the XML parsing and processing must occur for every user. So you should group as many users as possible with the same one policy, and then add the users to that policy. You should use the same LDAP group or role for as many users as possible in an organization.

Be sure that a policy is not modified or updated for each user. Modifying a policy is an expensive operation since the policy is stored as XML data.

Question: Is Sun Java Message Queue tuning necessary when session failover is configured for Access Manager?

Answer: In most deployments using Access Manager session failover, Java Message Queue tuning only requires setting adequate JVM heap and stack sizes. See the *Sun Java System Message Queue 2005Q1 Administration Guide* at `http://docs.sun.com/app/docs/coll/MessageQueue_2005Q1` for further information.

Question: When the `amtune` script tunes the Directory Server with the recommended values, an onscreen message says the tuning parameters such as `minConnPool` and `maxConnPool` in `serverconfig.xml` are dependent on the number of Access Manager instances and other factors. How exactly should I tune the Directory Server with these factors taken into account?

Answer: Values recommended by the `amtune` script for `minConnPool` and `maxConnPool` are per Access Manager server instance. The parameters are stored in `/etc/opt/SUNWam/config/serverconfig.xml`. The recommended values are based on the following assumptions:

- One AM server instance is in front of one Directory Server.

- The Directory Server contains both Access Manager configuration data and user data.

When multiple Access Manager instances exist, the total number of persistent LDAP connections may be too high for the Directory Server to handle. Each Access Manager instance establishes its own pool of the same size. Memory allocation is also on the high side if the user data is not stored there. The `amtune` script assumes the user data is stored together with Access Manager configuration data in the Directory Server.

For example, consider the typical real-world deployment scenario illustrated in the document *Deployment Example 1: Access Manager 7.1 Load Balancing, Distributed Authentication UI, and Session Failover*. This deployment differs from the `amtune` script assumptions in the following ways:

- Multiple Access Manager instances are in front of the Directory Server.
- The Directory Server instance stores only Access Manager configuration data, and does not store user data.
- A separate Directory Server stores the user data.

First, if you have a large number of Access Manager instances, you can reduce the recommended pool size for the configuration Directory Server. This only applies when you have large number of Access Manager instances. When you have only two or three Access Manager instances, it may not be necessary to reduce the pool size.

Secondly, you can significantly reduce the memory allocation to the configuration Directory Server . The configuration data is minimal with usually only a few thousands entries. Reducing the memory allocation is particularly important if the configuration Directory Server runs on the same host as the user data Directory Server. You do not want the smaller configuration Directory Server to compete with the larger user data Directory Server for the system memory.

Thirdly, be sure to tune the user data Directory Server. This directory contains a large data set. You can use the `amtune` recommended Directory Server tuning changes as a starting point. For more information, see step 3 of "Related Systems Tuning" on page 58.

Ultimately, you have to look at your directory data and tune it specifically. This is the standard Directory Server tuning procedure. See the *DS Performance Tuning Guide*.

Finally, the `amtune` script does not tune the LDAPv3 data store connection pool which is used by Access Manager to access the user data Directory Server. You have to manually tune the data store connection pool. See step 4 of "Related Systems Tuning" on page 58.

Question: Where do I find specific performance tuning guidelines for Access Manager implementations on the T2000 platform?

Answer: The Access Manager `amtune` script does the automatic tuning specifically for the T2000 platform. No manual tuning is necessary. The following is the tuning specific to T2000, done automatically by the `amtune` script.

Sun Fire CoolThreads technology servers, specifically Sun Fire T1000 and Sun Fire T2000 servers, contain a single Ultrasparc T1 chip or processor. The T1 processor is a unique design of 8 individual processing units, called cores, sharing one on-chip interconnection. It is somewhat like an 8-way system on a single chip.

Each core supports 4 hardware threads of execution. These hardware threads are scheduled on the core processing unit in round-robin order. A different software thread can run on each one of these hardware threads. Thirty-two software threads can run in parallel on a single T1 processor.

You can determine the number of cores by dividing the number of hardware threads (run `psrinfo -v`) by 4. The T1000 and T2000 can have a maximum of 4 hardware threads per core. So the number of cores is usually 6 (a 24 thread system) or 8 (a 32-thread system).

The only JVM parameter that would be different for Chip Multi-threading (CMT) servers is the following parameter

```
-XX:ParallelGCThreads=N
```

By default, if the parameter is not set, the value of `ParallelGCThreads` would be the same number as the number of hardware threads (either 24 or 32) on the T1000 and 2000. This is unnecessarily high. The `amtune` script today automatically sets the number of these parallel GC threads to be equal the number of cores in a T1000 or T2000 box.

For more information, see the document Java Tuning Whitepaper at `http://java.sun.com/performance/reference/whitepapers/tuning.html#section4.2.1` .

More Resources

For more information on performance tuning and troubleshooting, see the following resources:

- Java Performance portal site

  http://java.sun.com/javase/technologies/performance.jsp
- Java Tuning Whitepaper

  http://java.sun.com/performance/reference/whitepapers/tuning.html
- Java Hotspot VM Options

  http://java.sun.com/javase/technologies/hotspot/vmoptions.jsp
- Solaris TCP Tuning Parameters

  http://docs.sun.com/app/docs/doc/817-0404/6mg74vsaj?a=view
- Understanding Tuning TCP

  http://www.sun.com/blueprints/1205/819-5144.pdf
- Tuning for Linux platforms

  http://docs.sun.com/app/docs/doc/819-4742/6n6sfgme9?l=en&a=view
- Java 5.0 Troubleshooting and Diagnostic Guide

  http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf

PART III

Appendix

- Appendix A, “Known Issues and Workarounds”
- Appendix B, “Error Messages”

◆ ◆ ◆ APPENDIX A

Known Issues and Workarounds

The most common performance problems are identified by the following symptoms:

- "Memory Grows or Leaks" on page 81
- "System Responds Too Slowly" on page 81
- "Server Hangs and Does Not Respond" on page 83
- "Server Crashes" on page 87

Memory Grows or Leaks

- "WS 6.1 bundled with JES 4 fail to start" on page 81

WS 6.1 bundled with JES 4 fail to start

A memory leak may occur with securities libraries on Windows that were shipped with JES 4 for Windows. This causes the Web Server 6.1 (bundled with JES 4) to fail to start.

Solution: Install a standalone version of Weber Server 6.1 SP5 to force the Web Server to use its own bundled securities libraries from the `webserver/bin/https/bin` directory. Or you can use the JES 5 installer for installing Web Server 6.1SP5 or later.

System Responds Too Slowly

- "Application Server where Access Manager is deployed throws "cannot create thread" error" on page 82
- "Directory Server 5.2 hangs and shows high CPU usage when deleting entries" on page 82
- "Throughput performance of AM is significantly slower when it is deployed on WebLogic or WebSphere Application Server." on page 83

Application Server where Access Manager is deployed throws "cannot create thread" error

You see an error saying "Cannot create thread" with the following stack trace:

```
"Access ManagerSessionPoller[9]" daemon prio=10
tid=0x0985e2e0 nid=0x37 in Object.wait() [0x10519000..0x10519a38]
      at java.lang.Object.wait(Native Method)
      - waiting on <0x2ad92c18> (a java.util.ArrayList)
      at java.lang.Object.wait(Object.java:474)
      at com.iplanet.Access Manager.util.ThreadPool.getTask
          (ThreadPool.java:125)
      - locked <0x2ad92c18> (a java.util.ArrayList)
      at com.iplanet.Access Manager.util.ThreadPool$
          WorkerThread.run(ThreadPool.java:144)"
```

The problem is due to an insufficient amount of JVM heap size, or invalid Access Manager session threads are created out of control. This behavior is expected and not a deadlock at all.

Solution: To increase the JVM heap size, you can change the `domain.xml` manually or simply run Access Manager `amtune-as8`.

Directory Server 5.2 hangs and shows high CPU usage when deleting entries

You may see in the error log stating that "search is not indexed". The Directory Server referential integrity plug-in is automatically enabled by the Access Manager. But no indexes exist for Access Manager's attributes such as `iplanet-Access Manager-static-group-dn` and `iplanet-Access Manager-modifiable-by`. Access Manager does not configure the arguments of the plug-in, but uses the default arguments (`update interval=0`). Every deletion causes an immediate integrity check, which consumes a lot of system resources when the search is not indexed.

Solution: Be sure the referential integrity plug-in is enabled and configured, *and* that the attributes to be maintained are indexed. Be sure the referential integrity is configured to be executed synchronously or with a delay. A delay will remove the thread shortage per application.

If you observe that one or more of the deleted entries are repeatedly added or deleted, over time these entries may trigger non-indexed searches to the database. This issue is addressed in later versions of Directory Server. Upgrade to Directory Server 5.2 Patch 5 or Directory Server 6.0.

Throughput performance of AM is significantly slower when it is deployed on WebLogic or WebSphere Application Server.

Solution: First tune the JVM heap and GC (garbage collection) options for WebSphere Application Server. See "Third-Party Web Containers" on page 49 for more information. Since JVM 1.4.2 or earlier is much slower than JVM 1.5.0 or later in throughput performance, make sure the JVM version used in the container is 1.4.2 or later when you use the CMS and NewParallelGC options. WebLogic 9.0 or later and WebSphere 6.0 or later use JDK 1.5.0 or later.

"OutOfMemoryError" When Access Manager is deployed in WebLogic or WebSphere Application Server

This occurs on Access Manager 7.0 or higher, even with sufficient JVM heap sizes.

Solution: Be sure that in addition to having sufficient JVM heap size, the CMS and NewParallel GC options are specified. Also be sure that `-XX:-CMSParallelRemarkEnabled` is included. See "Third-Party Web Containers" on page 49 for more information.

Server Hangs and Does Not Respond

- "Access Manager Server hangs during session failover " on page 84
- "Server hangs when processing request between the load balancer and the Access Manager server. " on page 85
- "Access Manager server hangs when Sun Java System Directory Server restarts" on page 85
- "Access Manager unable to recover after a crash or watch dog restart under heavy load" on page 86
- " jaxrpc getAttributes throws SSOException" on page 86
- "Sun Java System Web Server hangs while handling a large number of images files" on page 86
- "Access Manager Web Policy Agent hangs" on page 86
- "Access Manager server hangs when multiple clients point to one Access Manager server instance " on page 87
- "System hangs when Access Manager clientsdk.jar and Access Manager server are in the same JVM instance " on page 87

Access Manager Server hangs during session failover

The problem occurs when both Access Manager server and JMQ (and even BDB) are installed in one machine. Both web server instances hang. A thread dump from the first web server instance shows all its threads are in `socketRead` operations.

```
waiting:

- java.net.SocketInputStream.socketRead0
(java.io.FileDescriptor, byte[], int, int, int) @bci=0,
pc=0xf75e0274, methodOop=0xf33a7aa8
(Compiled frAccess Managere; information may be imprecise)
```

A thread dump of the second web server instance shows the corresponding `writePacketNoAck` calls from `jmsclient`:

```
-com.sun.messaging.jmq.jmsclient.ProtocolHandler.writePacketNoAck
(com.sun.messaging.jmq.io.ReadWritePacket) @bci=7, line=235,
pc=0xf79a56b4, methodOop=0xf3650320 (Compiled frAccess Managere;
information may be imprecise)
-com.sun.messaging.jmq.jmsclient.ProtocolHandler.writeJMSMessage
(javax.jms.Message) @bci=565, line=1567, pc=0xf76bc190,
methodOop=0xf36533c8 (Compiled frAccess Managere)
-com.sun.messaging.jmq.jmsclient.WriteChannel.sendWithFlowControl
(javax.jms.Message) @bci=10, line=123, pc=0xf7825278,
methodOop=0xf3689e48 (Compiled frAccess Managere)
-com.sun.messaging.jmq.jmsclient.TopicPublisherImpl.publish
(javax.jms.Message) @bci=2, line=73, pc=0xf782ece0,
methodOop=0xf36b2400 (Compiled frAccess Managere)
-com.iplanet.dpro.session.jmqdb.JMQSessionRepository.save
(com.iplanet.dpro.session.service.InternalSession) @bci=92,
line=346, pc=0xf7775008, methodOop=0xf3604770 (Compiled frAccess Managere)
-com.iplanet.dpro.session.service.SessionService.saveForFailover
(com.iplanet.dpro.session.service.InternalSession) @bci=26,
line=2485, pc=0xf7005c34, methodOop=0xf35da5e8 (Interpreted frAccess Managere)
- com.iplanet.dpro.session.service.InternalSession.updateForFailover()
@bci=46, line=969, pc=0xf74a7c48, methodOop=0xf36c0420
(Compiled frAccess Managere)
```

Under a heavy load, the Access Manager server web container process will use most of the machine's CPU resources. Then JMQ and/or BDB with Access Manager `sessiondb` will not have sufficient CPU resources to process incoming requests. The first Access Manager server instance's threads carrying requests cannot write to the second Access Manager server instance with JMQ in the back because of the lack of CPU resources. Also the first Access Manager server instance will have its threads built up because of the backlog on the second instance due to the lack of processing on the part of JMQ and/or BDB for updating the session table.

Solution: Install JMQ and BDB on their own boxes, separate from Access Manager server machine.

Server hangs when processing request between the load balancer and the Access Manager server.

The problem occurs when using two Access Manager 7.0 SP5 servers with a load balancer in front. You may see a stack trace such as this:

```
" at sun.net.www.protocol.http.HttpURLConnection.getInputStream
(HttpURLConnection.java:961)
   - locked <0xf0898b88> (a sun.net.www.protocol.http.HttpURLConnection)
*     *at com.iplanet.services.comm.client.PLLClient.send(PLLClient.java:196)
   at com.iplanet.services.comm.client.PLLClient.send(PLLClient.java:115)"
```

Solution: The problem might be a result of having the server instances separated by a firewall. If this is your environment, move the server instances behind the same firewall.

The problem could be due to a misconfiguration in the Platform Server List. The stack trace shown above occurs when the Platform Server List is missing its associated site ID's and server instances are denoted by virtual host names. Here is an example of a misconfigured Platform Server List:

```
site list : http://hostname:80|11
server list : http://hostname1:80|01
                  http://hostname2:80|02
```

Configure your Platform Server List to include the server identifiers. In the following example, 11 is the server identifier.

```
site list : http://hostname:80|11
server list : http://hostname1:80|01|11
                  http://hostname2:80|02|12
```

Access Manager server hangs when Sun Java System Directory Server restarts

Connections between Access Manager server and Directory Server seem to close unexpectedly due to unsynchronized access to a shared variable. This is a known problem that is fixed in LDAP JDK 4.19

Solution: Upgrade to LDAP JDK 4.19. Download Patch 119725–04–1 from the following URL: `http://sunsolve.sun.com/search/document.do?assetkey=1-21-119725-04-1` The patch will work for Solaris 9 and 10. The same patch is used for both SPARC and x86 platforms.

Access Manager unable to recover after a crash or watch dog restart under heavy load

This problem occurs when using an LDAP v3 plug-in because the plug-in is being initialized more than once.

Solution: This known problem was fixed in both Access Manager 7.0 Patch 6 and Access Manager 7.1 Patch 1. Upgrade to one of these Access Manager versions.

jaxrpc getAttributes throws SSOException

A 403 error occurs. The problem occurs when session expiration notifications come between the SSO validation call and the call to fetch profile attributes on the agent side. The SSO validation is successful, but the profile attribute fetch fails with an `SSOException` from the server. This is the expected behavior. However, the SOAP client is not processing this exception properly, and is re-constructing. The client side calls wrap up this exception with `IdRepoException`. As a result the agent is not notified about the `SSOException`.

Solution:A fix was made in `amclientsdk`. This known problem was fixed in both Access Manager 7.0 Patch 6 and Access Manager 7.1 Patch 1. Upgrade to one of these Access Manager versions.

Sun Java System Web Server hangs while handling a large number of images files

The web container that hosts Access Manager hangs while handling a large number of image files. The following errors are display in the Web Server `errors.log`

```
IO error:
           "java.io.IOException: WEB8004: Error flushing the output stream
            java.io.IOException: WEB8001: Write failed "
```

These entries indicate that Access Manager server was not involved in the hang of the web server but instead the root cause was due to the an IO error.

Access Manager Web Policy Agent hangs

When a web policy agent hangs, it is usually due to misconfiguration of the web container where the agent is installed, or misconfiguration of the Access Manager web container on another host system. An Access Manager server may be running out of some resources due to, for example, a runaway number of invalid sessions.

Solution: Set the Web Policy Agent debug logging level to the finest level, `all:5`. Examine the logs to determine the exact cause.

Access Manager server hangs when multiple clients point to one Access Manager server instance

When multiple clients point to an Access Manager server instance, the session polling mechanism used prior to Access Manager 7.1, can cause the Access Manager server to hang. The older polling mechanism was based on caching time.

Solution: Upgrade to Access Manager 7.1. In the 7.1 version, the session polling mode is configurable. You can use on either caching or idle time mode. By default it is based on the idle time.

System hangs when Access Manager clientsdk.jar and Access Manager server are in the same JVM instance

The problem occurs when both a client application with Access Manager `clientsdk.jar` and Access Manager server are in the Access Manager JVM instance. When an JavaEE application tries to access IdRepo attributes for any identity, then Access Manager server can hang. The problem is due to the unnecessary synchronization block in `SMS ServiceConfigImpl`, `OrganizationConfigManagerImpl` and `ServiceConfigManagerImpl` in Access Manager SDK. This known issue was address in Access Manager 7.1.

Solution: Upgrade to Access Manager 7.1

Server Crashes

- “Access Manager web container crashes with "StackOverFlowError" errors ” on page 87
- “Apache Web Agent 2.2 on Linux crashes” on page 88
- “Access Manager crashes in SSL mode” on page 88
- “Customized JSP page causes Web Server to crash” on page 88
- “Application Server or Web Server crashes under a heavy load” on page 88

Access Manager web container crashes with "StackOverFlowError" errors

The problem is known to occur when Access Manager 7.0 is deployed a web server, and the "-Xss128k" JVM option is used with 64-bit JVM on the web container. The problem can occur

with any java application. For 64-bit JVM's, the minimum per thread stack size should be 256k, -Xss256k, or even 512k since 64-bit VM's default per thread stack size is 1 mb.

Solution: For 64-bit JVMs, the minimum per thread stack size should be 512k since the 64-bit JVM default per thread stack size is 1 mb. The 64-bit JVM support was introduced starting with Web Server 6.1 SP5. Application Server 8.1 and 8.2 do not support 64-bit JVM, but Application Server 9.1 will. Access Manager and its `amtune` scripts support 64-bit JVM starting with AM 7.0 Patch 5. For more information, see
`http://java.sun.com/docs/hotspot/threads/threads.html`

Apache Web Agent 2.2 on Linux crashes

Apache Web Server crashes when Web Agent is deployed.

Solution: There is no solution at this time. Running the Apache Server in multi-process mode (MPM) of compilation and runtime modes are not supported by our Apache Agent in Access Manager.

Access Manager crashes in SSL mode

The problem occurs When Access Manager server is configured in SSL mode and there is outbound SSL traffic from Access Manager server to another Access Manager server or Directory Server. In some rare situations where SSL socket calls don't get closed and queue up, the Access Manager server can crash if it is configured in the default NSS/JSS mode of SSL.

Solution: Upgrade to JSS 4.2.5.

Customized JSP page causes Web Server to crash

Web Server crashes when a customized JSP page for Access Manager is deployed in Sun Java System Web Server. The problem occurs when using a Web Server version prior to 6.1 SP8. The problem is due to a known problem in Sun Web Server that is unrelated to Access Manager server. If the JSP has calls to `HttpServlet.getScheme` or `HttpServlet.service`, the Web Server can crash.

Solution: Upgrade to Web Server 6.1 SP8.

Application Server or Web Server crashes under a heavy load

The problem is known to occur when using Sun Fire T1000/T2000 hardware (cool threads, Niagara boxes) to deploy Access Manager server with Sun Java System Application Server or Sun Java System Web Server.

Solution: Set the appropriate memory library the Application Server `asenv.conf` file or in the Web Server start script. For Application Server, in the `asenv.conf` file, replace `LD_PRELOAD=/usr/lib/libmtmalloc.so` with `LD_PRELOAD=/usr/lib/libumem.so`. For more information, see `http://www.sun.com/servers/coolthreads/tnb/applications_sunone.jsp`.

For Web Server in the start script, replace `LIBMTMALLOC=/usr/lib/libmtmalloc.so` with `LIBMTMALLOC=/usr/lib/libumem.so`..

Use the latest JDK 1.5 version, at least 1.5.0_08 or later when Sun Fire T1000/T2000 boxes are used for Access Manager server deployments.

◆ ◆ ◆ APPENDIX B

Error Messages

The following are Access Manager error messages you may encounter in log files for Access Manager, Web Server, Directory Server, Portal Server, or the Policy Agent host.

Error Log for the J2EE Policy Agent Application Server

```
thread dump from "kill -3" command shows hundreds of waiting threads like:
service-j2ee" daemon prio=10 tid=0x00d59180 nid=0x11e
waiting for monitor entry//at com.sun.identity.jaxrpc.SOAPClient.encodeMessage//-
waiting to lock <0x787e3ad0> (a com.sun.identity.jaxrpc.SOAPClient)
```

Description: Error message is found in the J2EE Policy Agent web container log.

Cause: This issue is caused by an unnecessary synchronization in the SOAP client Java class in `amclientsdk` for `encode` and `send` methods. During a load test of URL policy mode with J2EE policy agents, hundreds of threads can be waiting to lock on the `com.sun.identity.jaxrpc.SOAPClient.send` method.

Solution: Two related bugs (CR 6302120 and CR 6517760) were fixed in Access Manager 7.0 Patch 5 and Access Manager 7.1 Patch 1.

Web Policy Agent amAgent.log File

```
Error 30284:bfc093a0 all: Connection::read():
NSPR Error while reading data:-5961
```

Description: The Access Manager server is busy responding to all the previous requests from a web policy agent, and cannot respond to this particular request. Then the socket timeout happens on the web policy agent side, and the user will see this error message in the web policy agent `amAgent.log`.

Cause: The agent has timed out waiting for a response from the Access Mnager server.

Solution: Be sure the Access Manager server is properly tuned with values recommended in the `amtune` script. Also be sure that the web agent HTTP request parameters are properly tuned.

Web Policy Agent amAgent.log File

```
Error 19516:9088eb0 AM_SSO_SERVICE:SSOTokenService::getSessionInfo():
Error 18 for sso token ID
Error 21907:9088eb0 PolicyEngine: am_policy_evaluate:
InternalException in Service::initialize() with error message:
Naming query failed during service creation. and code:21"
```

Description: These errors occur during stress tests of Web Agent 2.2 for Apache Server 2.0.59 on RedHat Linux 3.0.

Cause: These errors mean that the SSO token has expired on the server side, but the agent is still sending the expired SSO token. In normal cases, if the web policy agent sees this error, it will redirect to the Access Manager login page. The Access Manager server becomes overwhelmed from all the incoming requests from the web policy agent. Other errors may occur:

```
Error 30284:bfc093a0 all: Connection::read():
NSPR Error while reading data:-5961
Error 30154:bfc093a0 all: Connection::read():
NSPR Error while reading data:-5961
Error 30054:bfc093a0 all: Connection::read():
NSPR Error while reading data:"
```

These errors occur because the web policy agent has timed out waiting for a response from the Access Manager server. During this load test the Access Manager server was so busy responding to all the previous requests, it failed to respond to this particular request. Then the socket timeout happens on the agent side and the user will see this error message.

Solution: Be sure the Access Manager server is properly tuned with `amtune` script recommended values. Also be sure that the web agent HTTP request parameters are properly tuned.

Access Manager amclientSDK File or Error Log for the J2EE Policy Agent

```
ERROR: Send Polling Error:com.iplanet.am.util.ThreadPoolException:
amSessionPoller thread pookdkdkdkl's task queue is full.
```

Description: These errors can occur after you deploy the Distributed Authentication UI web application, J2EE agents, or in any situation where you deploy the Access Manager client SDK on a client machine. There errors are seen in the J2EE Agent web container log or in the `amclientSDK` container log.

Cause: The client SDK polling threadpool size and threshhold are not sufficient for the number of incoming sessions.

Solution: If you have many concurrent sessions, add the following properties and values in either the `AMConfig.properties` file or the `AMAgents.properties` file:

- `com.sun.identity.session.polling.threadpool.size=10`
- `com.sun.identity.session.polling.threadpool.threshold=10000`

Access Manager amSession.log File

```
ERROR: Sending Notification Error:com.iplanet.am.util.ThreadPoolException:
amSession thread pool's task queue is full.
```

Description: These errors can occur when the number of incoming Access Manager sessions is more than the notification threadpool size and threshold can handle.

Cause: The `AMConfig.properties`default values for `com.sun.identity.session.notification.threadpool.size` and `com.sun.identity.session.notification.threadpool.threshold` are too low.

Solution: Increase the value for `notification.threadpool.size` to be three times the number of CPUs, or cores in case of Niagara boxes, where the Access Manager server is installed. Increase the value for `notification.threadpool.threshold` or the queue to be 30% of `maxSessions` in the `AMConfig.properties` file . If the same error still occurs, then an agent may not processing incoming requests efficiently, or some other bottleneck exists on the client side or on the Access Manager web container.

Access Manager amSession.log File

```
ERROR: Individual notification to
http://mycompany.com:7001/agentapp/notificationcom.iplanet.services.
comm.server.SendNotificationException:
Server returned HTTP response code: 503 for
URL: http://yourcompany.com:7001/agentapp/notification
```

Description: These errors occur during a high load scenario when a bottleneck in the notification queue exists on the port between the Access Manager server and its policy agent machines.

Cause: The notification attempts to the agent were not successful.

Solution: There is no solution for this now. But starting with Federated Access Manager 8.0, notification traffic will use JMQ asynchronous publish and subscribe mechanisms with different ports, which will eliminate this kind of bottleneck.

Access Manager amComm.log File

```
ERROR: Cannot send notification to
http://mycompany.com:80/amagent/UpdateAgentCacheServlet?shortcircuit=
false java.io.IOException:Server returned HTTP response
code: 503 for URL:
http://yourcompany.com:80/amagent/UpdateAgentCacheServlet?shortcircuit=false
```

Description: These errors occur during a high load scenario when a bottleneck in the notification queue exists on the port between the Access Manager server and its policy agent machines.

Cause: The notification attempts to the agent were not successful.

Solution: There is no solution for this now. But starting with Federated Access Manager 8.0, notification traffic will use JMQ asynchronous publish and subscribe mechanisms with different ports, which will eliminate this kind of bottleneck.

Policy Agent amAgent.log File

```
Info PolicyAgent: am_web_result_attr_map_set():
No profile or session or response attributes to be set as headers or cookies
Debug all: Log::pSetLevelsFromString():
setting log level for module 0 to 4, old level 1.
```

Description: This error means the session or response attribute is missing from URL string. The Web Server crashes.

Cause: Insufficient size of the maximum length on the URL (query string length) that can be passed to web policy agent.

Solution: Upgrade to Web Policy Agent 2.2-HP8.

SAML2 Debug Log

This log is stored in the following location:

```
/var/opt/SUNWam/fm/federation/debug/fmSAML2

ERROR: Unable to send SOAPMessage to IDP
com.sun.xml.messaging.saaj.SOAPExceptionImpl:
java.security.PrivilegedActionException:
com.sun.xml.messaging.saaj.SOAPExceptionImpl: Unable to internalize message
at com.sun.xml.messaging.saaj.client.p2p.HttpSOAPConnection.call
at com.sun.identity.saml2.common.SAML2Utils.sendSOAPMessage
at com.sun.identity.saml2.profile.LogoutUtil.doSLOBySOAP
```

Description: In a high availability and high load scenario, for example when more than one Access Manager server or Federation Manager are behind a load balancer, the SOAP Global logout fails if it redirects to the wrong server.

Cause: The signature string is not forwarded when redirected to the internal server instance.

Solution: This bug was fixed in SAML v2 Patch 3. The fix delays the signature validation until the Access Manager or Federation Manager server finds the session in the local server. This way there is little processing involved before the signature verification is done. Update to SAML v2 Patch 3.

SAML2 Debug Log

This log is stored in the following location:

```
/var/opt/SUNWam/fm/federation/debug/fmSAML2

ERROR: Unable to get infoKeyString from SSOToken.
ERROR: Error sending Logout Request
com.sun.identity.saml2.common.SAML2Exception:
Error retrieving NameIdInfoKey from SSOToken.
at com.sun.identity.saml2.profile.SPSingleLogout.initiateLogoutRequest
at _jsps._saml2._jsp._spSingleLogoutInit_jsp._jspService
```

Description: This error will sometimes occur during a high load scenario even with SAML v2 Patch3. The `NameIDInfoKey` information is stored in session properties, but sometimes during a high load scenario, the information cannot be retrieved.

Cause: The session properties do not get refreshed immediately.

Solution: Refresh the session properties before reading the `NameIDInfoKey`.

Error Log for Web Server or Application Server

```
SEC_ERROR_NO_MEMORY: Out of memory
```

Description: Web Server or Application Server process crashes. This crash will occur only if SSL is enabled for the Web Server or Application Server.

Cause: A bug exists in NSPR 4.5. The NSPR threads created in linux use 10240kb as the stack size regardless of the stack size specified during thread creation. The default is 10240 kb per thread stack on the Red Hat Linux platform.

Solution: Upgrade the NSPR version to 4.6.

Error Log for Web Server or Application Server

```
Cannot create thread.
amSessionPoller [9] daemon prio=10
tid=0x0985e2e0 nid=0x37 in Object.wait () [0x10519000..0x10519a38
at java.lang.Object.wait (Native Method)
```

Description: Web Server or Application Server processes cannot create any more threads for Access Manager sessions.

Cause: Insufficient JVM heap size or invalid Access Manager session threads are created out of control. This behavior is expected.

Solution: To increase the JVM heap size, change the `domain.xml` manually or run the Access Manager `amtune-as8` script.

Error Log for Web Server or Application Server

```
java.IOException:Not enough space
at java.lang.UNIXProcess.forkAndExec(Native Method)
at java.lang.UNIXProcess.forkAndExec
at java.lang.UNIXProcess
```

Description: The JVM cannot launch itself while trying to fork a process through the system.

Cause: Either there are not enough file descriptors, or there is not enough swap space.

Solution: Do one of the following:

- Increase the number of system file descriptors, then reboot the machine. To increase the number of file descriptors, you can run the `amtune-os` script, manually set them by running the command `ulimit -n` *number_of_file_descriptors*.
- Increase the swap space by killing unnecessary processes.
- Add more swap space using `swap` command.

Error Log for Web Server or Application Server

```
Exception in thread "service-j2ee" java.lang.OutOfMemoryError:
requested 53515 bytes for jbyte in
/BUILD_AREA/jdk1.5.0_10/hotspot/src/share/vm/prims/jni.cpp.
Out of swap space?
```

Description: The native heap allocation failed and the native heap may be close to exhaustion.

Cause: A native code leak, for example the C or C++ code, continuously requires memory without releasing it to the operating system. There could be indirect causes like an insufficient amount of swap space or another process that is consuming all memory or leaking it.

Solution: For further diagnosis of a native code memory leak, see the *Java 5.0 Troubleshooting and Disgnostic Guide* at `http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf`. In the section "Diagnosing Leaks in Native Code." See the information about tools for different operating systems. The tools include `mdb` and `dbx` (runtime `trace`) for Solaris 9 U3 or later, `mtrace` , `libnjamd` for Linux, and `windb` or `userdump` for Windows."

Error Log for Web Server or Application Server

```
Exception in thread "main" java.lang.OutOfMemoryError: <reason>
<stack trace>(Native method)
```

Description: A native method has encountered a memory allocation failure. The difference between this and the previous error message is that the allocation failure here is detected in a JNI or native method rather than VM code.

Cause: A native code leak, for example the C or C++ code, continuously requires memory without releasing it to the operating system. There could be indirect causes like an insufficient amount of swap space or another process that is consuming all memory or leaking it.

Solution: For further diagnosis of a native code memory leak, see the *Java 5.0 Trouble-Shotting and Disgnostic Guide* section "Diagnosing Leaks in Native Code."

Error Log for Web Server or Application Server

```
Exception in thread ?main? java.lang.OutOfMemoryError: Java heap space
```

Description: An object could not be allocated in the Java heap.

Cause: The cause is a simple configuration issue. The maximum heap size, noted by `-mx options` is not sufficient for the load. Or the application may be holding references to objects which cannot be garbage collected. This is a Java equivalent of a memory leak. If the `finalize` method is used so much that is that the finalizer daemon thread cannot keep up with the finalization queue, then this error can occur when the heap becomes full.

Solution: Maximum JVM option increase or coding changes.

Error Log for Web Server or Application Server

```
Exception in thread ?main? java.lang.OutOfMemoryError: PermGen space
```

Description: This error occurs when there are a large number of class, method or String objects.

Cause: The permanent generation is full. The permanent generation is the area of the heap where class and method objects are stored and `java.lang.String` objects are interned.

Solution: The JVM option for Perm size may need to be increased.

Error Log for Web Server or Application Server

`Exception in thread "main" java.lang.StackOverflowError at java.lang.String.indexOf`

Description: The JVM (java) stack size is not sufficient

Cause: There can be many types of `StackOverflowError` errors including a wrong server instance name in the platform list on the Access Manager console, or any one of numerous Java coding issues. But here the only type of `StackOverflowError` that will be addressed is the one that can occur when you use 64-bit JVM with the `-Xss128k` option.

Solution: For 64-bit JVM's, the minimum per thread stack size should be at least 256k, -Xss256k, or even 512k since 64-bit VM's default per thread stack size is 1 mb. 64-bit JVM support was introduced starting with Web Server 6.1 SP5 or later, including Web Server 7.0. Application Server 8.1 and 8.2 do not support 64-bit JVM, but Application Server 9.1 will. Access Manager and its `amtune` scripts support64-bit JVM, starting with AM 7.0 Patch 5 and 7.1. For more information, see `http://java.sun.com/docs/hotspot/threads/threads.html`.

Index

A

Access Manager
  caches, 26
  instance name, 31
  multiple instances, 31
  running tuning scripts, 22
  tuning modes, 21
  tuning parameters, 24
  tuning scripts, 19
Access Manager Administrator password, 22
admin host name, Application Server 8, 34
admin password, Application Server, 22
admin port, Application Server 8, 34
admin utility location, Application Server 8, 34
administrator user account, Application Server 8, 34
AMConfig.properties file, 21, 25, 31
amtune-as7 script, 20
amtune-as8 script, 20
AMTUNE_DEBUG_FILE_PREFIX parameter, 25
AMTUNE_DONT_TOUCH_SESSION_PARAMETERS
  parameter, 27
amtune-env file
  description of, 23
  editing, 22
amtune-identity script, 20
AMTUNE_MODE parameter, 21, 24
AMTUNE_PCT_MEMORY_TO_USE parameter, 26
AMTUNE_PER_THREAD_STACK_SIZE
  parameter, 27
amtune-prepareDSTuner script, 20
amtune script
  description of, 20
amtune script *(Continued)*
  running, 23
AMTUNE_SESSION_MAX_CACHING_TIME_IN_MTS
  parameter, 28
AMTUNE_SESSION_MAX_IDLE_TIME_IN_MTS
  parameter, 28
AMTUNE_SESSION_MAX_SESSION_TIME_IN_MTS
  parameter, 28
AMTUNE_TUNE_DS parameter, 24, 38
AMTUNE_TUNE_IDENTITY parameter, 24
AMTUNE_TUNE_OS parameter, 24
AMTUNE_TUNE_WEB_CONTAINER
  parameter, 24
AMTUNE_WEB_CONTAINER_JAVA_POLICY
  parameter, 35
amtune-ws61 script, 20
Application Server
  admin password, 22
  tuning parameters, 33
  tuning script for, 20
ASADMIN_HOST parameter, 34
ASADMIN_INTERACTIVE parameter, 34
ASADMIN parameter, 34
ASADMIN_PASSFILE parameter, 34
ASADMIN_PORT parameter, 34
ASADMIN_SECURE parameter, 34
ASADMIN_TARGET parameter, 34
ASADMIN_USER parameter, 34

B
base directory, web container, 30

C
CHANGE mode, 21, 23, 24, 40
com.iplanet.am.notification.threadpool.size parameter, 26
com.iplanet.am.notification.threadpool.threshold parameter, 26
com.iplanet.am.session.httpSession.enabled parameter, 26
com.iplanet.am.session.invalidsessionmaxtime parameter, 26
com.iplanet.am.session.maxSessions parameter, 26
com.iplanet.am.session.purgedelay parameter, 26
com.iplanet.services.debug.directory parameter, 21, 25
CONTAINER_BASE_DIR parameter, 30
CONTAINER_INSTANCE_DIR parameter, 32

D
debug log file, checking, 23, 40
DEFAULT_ORG_PEOPLE_CONTAINER parameter, 38
Directory Manager password, 22
Directory Server
tuning, 37
tuning parameters, 37
DIRMGR_UID parameter, 38
documentation
collections, 12
related Java ES product, 12
domain name, specifying, 29
domainname command, 29
DOMAINNAME parameter, 29

H
host name, specifying, 29
hostname command, 29
HOSTNAME parameter, 29

I
installation directory, tuning scripts, 19
instance name, Access Manager, 31
IS_INSTANCE_NAME parameter, 31

J
Java security descriptors, 35
JVM memory usage, 26

L
Linux systems
base directory, 19
tuning scripts for, 19

M
maximum number of cache entries, 26
maximum number of sessions, 26
maximum session cache time, tuning, 28
maximum session idle time, tuning, 28
maximum session time, tuning, 28
modes, tuning, 21
multiple instances, Access Manager, 31

P
password, Access Manager Administrator, 22
password, Directory Manager, 22
password file location, Application Server 8, 34
Portal Server, 34

R
RAM, used by Access Manager, 26
RAM_DISK parameter, 38
restart, required during tuning, 23
REVIEW mode, 21, 23, 24, 39
root, running scripts as, 22, 39

S

SDK caches, 26
server.policy file, 35
session caches, 26
session time-out tuning, 27
Solaris system
  tuning OS kernel, 24
  tuning scripts for, 19
Sun Java System Access Manager, 19
Sun Java System Application Server, tuning script for, 20
Sun Java System Portal Server, 34
Sun Java System Web Server, tuning script for, 20
superuser, running scripts as, 22, 39
syntax to run tuning scripts, 22

T

TCP/IP settings, tuning, 24
thread pool sizes, 26
tuning modes, 21
tuning parameters
  Access Manager, 24
  Application Server 8, 33
  Directory Server, 37
tuning scripts
  description of, 19
  running, 22
  syntax to run, 22

W

web container, instance name, 30
Web container, specifying for tuning, 30
WEB_CONTAINER_INSTANCE_NAME parameter, 30
WEB_CONTAINER parameter, 22, 30
Web Server, tuning script for, 20