# Sun OpenSSO Enterprise 8.0 Performance Tuning Guide

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

Part No: 820–3747–12
January 2010

100126@23031

# Contents

CHAPTER 1

# 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 7
- “Using a Systematic Approach to Performance Tuning” on page 8

## Avoiding Common Performance Testing and Tuning Mistakes

The following is a common approach that is fraught with peril. Deployment engineers construct the system and perform the functional tests. Next the engineers hand over the entire 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.

The testing team then realizes that performance tuning was not done before the tests were run. 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.

A deployment often consists of a half-dozen or more systems with complex behaviors. By testing the entire system directly and troubleshooting problems as they occur, you can unnecessarily complicate the issue resolution process. Even a trivial problem in lower-level components takes great effort to identify, isolate, and address. Problems buried even further down may not be observable at a high level, and can remain buried until discovered in production.

In application development, this approach would be similar to conducting system integration tests without conducting unit tests. Every application developer knows how counter-productive it is to skip unit tests. It's not surprising to see that when conducting performance tests, it is also counter-productive to skip unit tests.

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

First, you should break down the entire system into smaller, independent components. Each component should be performance-tested independently or in a smaller group context. Develop special test scripts to unit-test the components. Next, combine unit-tested components together in slightly larger functional groups, and then test them in this context. Again, you may need to develop special test scripts and special setups. Finally, test the entire system as a whole.

This systematic approach requires early involvement and cooperation from both the performance team and the application development team. The performance team must have adequate knowledge about the deployment to execute the system breakdown tests. The best practice is a systematic approach to performance testing with an allocation of a minimum of three weeks testing time in addition to the early project involvement.

The following sections illustrate how to properly execute such performance tests.

1. “Constructing the System” on page 8
2. “Automated Performance Tuning” on page 9
3. “Related Systems Tuning” on page 9
4. “Baseline Modular Performance Testing” on page 10
5. “Advanced Performance Tuning” on page 17
6. “Targeted Performance Testing” on page 18

## 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: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0*. Each module 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 “Configuring Load Balancer 2 for OpenSSO Enterprise” in *Deployment Example: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0*

## Automated Performance Tuning

In this phase, you tune the system using the amtune tool that comes with the product. The amtune tool automates most of the performance tunings and address most, if not all, OpenSSO Enterprise 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 OpenSSO Enterprise system. For more detailed information, see "About the amtune Tool" on page 19.
2. Follow the amtune onscreen prompts to tune the OpenSSO configuration directory server instances (if it's not OpenDS). 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. If the OpenSSO Enterprise sub-realm is pointing to an external Directory Server user database, then manually tune the sub-realm LDAP connection pool.

   The amtune tool tunes only the LDAP connection pools of the root realm. You can configure the following parameters on LDAPv3 IDrepo:

   a. LDAP Connection Pool Minimum Size
   b. LDAP Connection Pool Maximum Size
4. 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 OpenSSO Enterprise is deployed on a Sun Web Server, the amtune tool 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.
5. If you have installed a J2EE Policy Agent on an application server, see "Tuning Third-Party Containers" on page 24 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 tool. 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 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 clearly 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% OpenSSO Enterprise 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 OpenSSO Enterprise instance are used. For your tests, you could use 200 users for one OpenSSO Enterprise instance, 400 users for two OpenSSO Enterprise instances, 600 users for three OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise, 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 systems.

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

```
cd /opt/SUNwbsvr/https-<openssoHost>/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 OpenSSO Enterprise directly.
2. Log in and time out on each individual OpenSSO Enterprise directly.
3. Log in and log out using a load balancer with one OpenSSO Enterprise server.
4. Log in and time out using a load balancer with one OpenSSO Enterprise server.
5. Log in and log out test one load balancer with two OpenSSO Enterprise instances behind.
6. Perform login and timeout test one load balancer with two OpenSSO Enterprise instances behind.

If you have two OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise instance. If there is a large variance in throughput, investigate why one server behaves differently than another.

**Compare the maximum authentication throughput of individual OpenSSO Enterprise 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 OpenSSO Enterprise instances is roughly twice the throughput on a load balancer with one OpenSSO Enterprise instance behind it.**

If the throughput numbers do not increase proportionately with the number of OpenSSO Enterprise instances, you have not configured sticky load balancing properly. Users logged in to one OpenSSO Enterprise instance are being redirected to another instance for logout. You must correct the load balancer configuration. For related information, see “Configuring Load Balancer 2 for OpenSSO Enterprise” in *Deployment Example: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0*.

**Verify that for each test, the OpenSSO Enterprise transaction counts report indicates no unexpected OpenSSO Enterprise requests.**

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

```
# cp access a; grep Login a|wc; grep naming a|wc; grep sesion a|wc;
grep policy a|wc; grep jaxrpc a|wc; grep notifi a|wc; grep Logout a|wc;wc a;
    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 OpenSSO Enterprise 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 OpenSSO Enterprise 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:

```
# cp access a; grep Login a|wc; grep naming a|wc; grep sesion a|wc;
grep policy a|wc; grep jaxrpc a|wc; grep notifi a|wc; grep Logout a|wc;wc a;
    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 OpenSSO Enterprise instance, then is sent to another OpenSSO Enterprise instance for logout. The second OpenSSO Enterprise instance must generate an extra session request to the originating OpenSSO Enterprise 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 OpenSSO Enterprise instances cannot double the throughout of the single OpenSSO Enterprise 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 that 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 to 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 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 OpenSSO Enterprise instance, with no load balancer in place.

  This test determines the OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise instances behind it.

  This test determines the baseline results when multiple OpenSSO Enterprise 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% OpenSSO Enterprise 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 OpenSSO Enterprise instance can be used . For example, you could use 200 users for one OpenSSO Enterprise instance, 400 users for two OpenSSO Enterprise instances, 600 users for three OpenSSO Enterprise 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 OpenSSO Enterprise instance is determined, continue to use this number for the rest of the tests to make the results more comparable. If you have two OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise 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, OpenSSO Enterprise does not reach full capacity, and the agent machine reaches full capacity first. In this case, since the agent machine is the bottleneck, and not the OpenSSO Enterprise machine, you can expect both the first and second page access to be slow while OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise instance. If there is a large variance in throughput, investigate why one server behaves differently than another.

**Compare the maximum authorization throughput of individual OpenSSO Enterprise 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 OpenSSO Enterprise instances is roughly twice the throughput on a load balancer with one OpenSSO Enterprise instance behind it.**

If the throughput numbers do not increase proportionately with the number of OpenSSO Enterprise instances, you have not configured sticky load balancing properly. Users logged in to one OpenSSO Enterprise instance are being redirected to another instance for logout. You must correct the load balancer configuration. When sticky load balancing is properly configured, each OpenSSO Enterprise should serve requests independently and thus the system would scale near linearly. If the throughput numbers do not increase proportionately with the number of OpenSSO Enterprise instances, you have not configured sticky load balancing correctly. For related information, see "Configuring Load Balancer 2 for OpenSSO Enterprise" in *Deployment Example: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0*.

**Verify that for each test, the OpenSSO Enterprise transaction counts report indicates no unexpected OpenSSO Enterprise requests.**

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

```
# cp access a; grep Login a|wc; grep naming a|wc; grep sesion a|wc;
grep policy a|wc; grep jaxrpc a|wc; grep notifi a|wc; grep Logout a|wc;wc a;
    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 OpenSSO Enterprise 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 OpenSSO Enterprise 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:

```
# cp access a; grep Login a|wc; grep naming a|wc; grep sesion a|wc;
grep policy a|wc; grep jaxrpc a|wc; grep notifi a|wc; grep Logout a|wc;wc a;
    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 five session requests, and 0.75 notifications. The total number of requests do add up though. This indicates there are no other unexpected requests. There are more session requests per login because the sticky load balancing is not working properly. A user logged in on one OpenSSO Enterprise instance is sometimes sent to another OpenSSO Enterprise instance for session validation and logout. The second OpenSSO Enterprise instance must generate extra session and notification requests to the originating OpenSSO Enterprise 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 OpenSSO Enterprise instances cannot double the throughout of the single OpenSSO Enterprise 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` tool 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. 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` tool. 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 to give you both confidence that the system performs properly, and a rough idea of how the system should perform in your targeted performance test scenarios. Targeted 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 2

# Tuning Components in the OpenSSO Enterprise Deployment

This chapter provides instructions for installing and using the amtune tool to tune OpenSSO Enterprise and related components. The following topics are contained in this chapter:

- "About the amtune Tool" on page 19
- "Using the amtune Tool" on page 25

## About the amtune Tool

- "Using a Password File" on page 20
- "Using amtune Modes" on page 21
- "Tuning the Operating System" on page 21
- "Tuning Third-Party Containers" on page 24

The OpenSSO Enterprise amtune tool enables you to tune the major components of your deployment. In previous versions of OpenSSO (known as Sun Java System Access Manager), a collection of amtune shell scripts was used to do the tunings. In OpenSSO Enterprise, the amtune tool is a Java application. Before you can use the amtune tool, the following conditions must be met:

- OpenSSO Enterprise must be deployed on a supported web container.

  The following are supported web containers: Geronimo 2.x, JBoss 4.2 and 5.1, Oracle Application Server 10g, Tomcat 5 and 6, WebLogic 9.2 MP2 and 10.3, WebSphere 6.1 and 7, Sun Web Server 7, and Sun Application Server 9.1 and GlassFish v2.
- The host computer must be running a supported operating system.

  The following are supported operating systems: Solaris 9 and 10, Red Hat Enterprise 4 and 5, Windows 6.1 and 7, Ubuntu 8.04 or later for some containers, and IBM AIX 5.3 for only WebSphere 6.1 and 7.

The following table lists the components tuned by the amtune tool.

**TABLE 2–1** Components Tuned by the amtune Tool

| Component | What Gets Tuned |
|---|---|
| OpenSSO Enterprise | ■ Session entries<br>■ Logging buffer<br>■ Notification thread pool/queue<br>■ LDAP connection pool sizes for service management, global authentication service and user data store |
| Sun Web Server 7<br>Sun Application Server 9.1 and GlassFish v2 | ■ JRE heap and JRE per-thread stack sizes<br>■ JVM garbage collection algorithms<br>■ Container worker and acceptor threads and queue sizes |
| Sun Java System Directory Server 5.2 and 6.3 | ■ Worker threads<br>■ Database cache and entry cache sizes<br>The `amtune` tool changes the database home directory to a RAM disk location such as `/tmp`. |
| Other | ■ The operating system (OS) kernel<br>Solaris and Linux platforms only.<br>■ TCP parameters |

The `amtune` tool relies on a list of DO NOT MODIFY parameters in the last section of the `amtune-env.properties` file. The parameters in that section are mainly for internal use by `amtune`. Do not modify the parameters in the DO NOT MODIFY list unless user tests show significant improvement in performance.

# Using a Password File

Execute `amtune` or `amtune.bat` with a file that contains passwords for the servers in your deployment. Use the following strings:

| Server | String |
|---|---|
| Sun OpenSSO Enterprise 8.0 | SSOADM_PASSWORD= |
| Sun Web Server 7.0 | WADM_PASSWORD= |
| Sun Application Server 9.1 | ASADMINPASSWORD= |
| Sun Directory Server | DIRMGR_PASSWORD=_ |

For sample entries, see the sample passwords file. The file `amtune-samplepasswordfile` is located in the directory:

*TOOLS_DIR*\*OPENSSO_URI*`\bin\amtune`,

where `amtune/amtune.bat` and `amtune-env.properties` are also located.

On Solaris, Linux, and AIX, the password file must be inaccessible to non-owners and only readable by its owner . For example, you can change the permissions mode of the password file by running the following command:

```
chmod 400
```

On Windows, `amtune.bat` does not check the permission on the password file.

Tuning operating system parameters does not require a password file.

## Using amtune Modes

You can run the `amtune` tool in REVIEW mode (the default) or in CHANGE mode, as determined by the `AMTUNE_MODE` parameter in the `amtune-env.properties` file.

REVIEW — This is the default value. Returns tuning recommendations for an OpenSSO Enterprise deployment, but does not make any actual changes to the environment.

CHANGE — Makes all tuning modifications defined in the `amtune-env.properties` file. Use CHANGE mode only after you have reviewed and understand the tuning changes that will be applied to your deployment.

In either mode, the tool returns a list of tuning recommendations to the terminal window and to the following log file:
`<TOOLS_DIR>\<OPENSSO_URI>\logs\amtune-config.<timestamp>.log`

Any error messages due to missing or invalid data in the `amtune-env.properties` file are displayed in the terminal window and written to the following file: `<TOOLS_DIR>\<OPENSSO_URI>\logs\amtune-errors`. All other error messages triggered by underlying components such as OpenSSO Enterprise or Sun Web Server are also written to the `amtune-errors` file.

## Tuning the Operating System

The `amtune` tool tunes the operating system parameters only on Solaris and Linux. It does not tune the operating system parameters on AIX, Windows, MacOS or BSD variants.

## Linux OS

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

- "File Descriptors" on page 22
- "TCP Settings" on page 23

For detailed information on tuning Linux operating system parameters, see the *IBM Linux Performance and Tuning Guidelines*.

### File Descriptors

You might need to increase the number of file descriptors from the default. 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 survive a system reboot, add it to `/etc/sysctl.conf` and specify the maximum number of open files permitted:

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

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:

```
ulimit -a
```

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 open files 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 can also specify a user ID instead.

## TCP 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 60 > /proc/sys/net/ipv4/tcp_keepalive_time
         echo 75 > /proc/sys/net/ipv4/tcp_keepalive_intvl
         echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
   ```

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

   ```
         net.ipv4.ip_local_port_range = 1204 65000
         net.core.rmem_max = 8388608
         net.ipv4.tcp_rmem = 4096 131072 8388608
         net.ipv4.tcp_wmem = 4096 131072 8388608
         net.ipv4.tcp_sack = 0
         net.ipv4.tcp_timestamps = 0
         net.ipv4.tcp_window_scaling = 0
         net.ipv4.tcp_keepalive_time = 60
         net.ipv4.tcp_keepalive_intvl = 75
         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.

## Tuning JRE Heap Sizes

The amtune tool restarts the server to check its JVM mode and to determine how much heap size is available for setting OpenSSO cache and session entries. For other web containers, the amtune tool supports only 32-bit JRE. For other web containers, $WEB_CONTAINER and $CONTAINER_INSTANCE_DIR values are not required.

Although the amtune tool does not tune non-Sun web containers, it will tune OpenSSO parameters if $AMTUNE _TUNE_OPENSSO is set to true.

By default, the amtune tool runs based on the assumption that the following amount of memory (megabytes) is available for tuning OpenSSO when the web container (both Sun and non-Sun) is running with 32-bit JRE:

- (Sparc/x86/AIX) AMTUNE_MAX_MEMORY_TO_USE_IN_MB_SOLARIS=3584
- (Linux) AMTUNE_MAX_MEMORY_TO_USE_IN_MB_X86=2341
- (Windows) AMTUNE_MAX_MEMORY_TO_USE_IN_MB_DEFAULT=1536

The amtune tool also tunes OpenSSO Enterprise when it is deployed on WebSphere 6.1 and 7, and on AIX, although it does not tune IBM AIX system parameters or WebSphere container parameters.

For 64-bit JRE, the amtune tool limits the initial heap size (-Xms) to 12 GB for Web Server 7, and for Application Server 9.1 and GlassFish v2. If the Solaris operating system has at least twice as much virtual memory (swap space) as the desired initial JVM heap size, the initial heap size can be increased manually. There is no limit for the maximum heap size (-Xmx).

Using 64–bit JRE, the user session cache size and number of sessions are calculated by the amtune tool. The results can be many times more than those calculated for 32–bit JRE, depending upon available memory. Be sure to review these numbers and determine whether or not they are apropriate.

## Tuning Third-Party Containers

For 32-bit Sun JRE 1.5 on Solaris 10 (both Sparc and x86), the following JVM options can be used as an example. The actual heap sizes should be adjusted based on the available physical memory, other processes running and the presence of any other active web applications running in the same JVM as OpenSSO.

```
-server
-XX:+UseParNewGC
-XX:+UseConcMarkSweepGC
-Xms3136M
-Xmx3136M
-XX:NewSize=392M
```

```
-XX:MaxNewSize=392M
-Xss128k
```

If JRE 1.6 is used, the following diagnostic JVM options can be added:

- `-XX:+HeapDumpOnOutOfMemoryError`
- `-XX:+PrintConcurrentLocks`

For more information on troubleshooting JRE 6 deployment, see "Troubleshooting Java SE 6 Deployment".

For more efficient garbage collection processing of soft reference objects, add the `-XX:+DoEscapeAnalysis` JVM option which is available with JDK 1.6.0_14 and later versions. To improve the performance of 64–bit JRE, add the `-XX:+UseCompressedOops` JVM option which is available with JDK 1.6.0_14 and later versions. This option compresses object references to 32 bits of the 64–bit JRE heap if the total heap is less than 32GB in size, reducing the amount of data that the HotSpot garbage collection engine must process.

---

**Note –** The Escape analysis-based optimization option `-XX:+DoEscapeAnalysis` is disabled in JDK 1.6.0_18. This option will be restored in a future Java SE 6 update. For more information on these two options, see "Java SE 6 Update Release Notes."

---

For WebLogic Application Server, increase the `MaxPermSize` from the default value of 128m to 256m in `setDomainEnv.sh` as shown below:

```
"if [ "${JAVA_VENDOR}" = "Sun" ] ; then
       MEM_ARGS="${MEM_ARGS} ${MEM_DEV_ARGS} -XX:MaxPermSize=256m"
       export MEM_ARGS
fi"
```

Otherwise, WebLogic Application Server may not start up with OpenSSO Enterprise 8.0 deployed.

# Using the `amtune` Tool

- "To Install the `amtune` Tool" on page 26
- "To Tune the Operating System, Web Container, and OpenSSO Enterprise" on page 26
- "To Tune a Remote Sun Directory Server" on page 27

## ▼ To Install the amtune Tool

1. **In the directory where you want to install the tuning tool, unzip the `ssoAdmintools.zip` file.**

2. **In the directory which has the unzipped `ssoAdminTools` file, run the following command:**

   |  |  |
   |---|---|
   | UNIX | `./setup` |
   | Windows | `setup` |

3. **When prompted, enter the configuration directory.**

   Example: `/opensso_config/opensso`

   Once installation is complete, you can use the command-line interface under the following directory:

   ```
   <TOOLS_DIR>/<OPENSSO_INSTANCE_NAME>/bin
   ```

   where `<TOOLS_DIR>` is the directory which has the unzipped `ssoAdmintools.zip` file, and `<OPENSSO_INSTANCE_NAME>` is the OpenSSO Enterprise deployment URI.

   The amtune tool is now in the `<TOOLS_DIR>/<OPENSSO_URI>/bin/amtune` directory.

## ▼ To Tune the Operating System, Web Container, and OpenSSO Enterprise

1. **Log in as or change to superuser.**

2. **If you have not run the tool in REVIEW mode, ensure that AMTUNE_MODE is set to REVIEW in the following file:**

   ```
   <TOOLS_DIR>\<OPENSSO_URI>\bin\amtune\amtune-env.properties file
   ```

3. **Edit other parameters in the `amtune-env.properties` file, depending upon the components you want to tune.**

   See the Appendix for detailed information about the properties.

4. **In REVIEW mode, run the `amtune` tool with a password file.**

   See the section "Using a Password File" on page 20 in this document.

   |  |  |
   |---|---|
   | Solaris or Linux | `<TOOLS_DIR>/<OPENSSO_URI>/bin/amtune/amtune` |
   | Windows | `<TOOLS_DIR>\<OPENSSO_URI>\bin\amtune\amtune.bat` |

Review the tuning recommendations in the terminal or in the `<TOOLS_DIR>/OPENSSO_URI/logs/amtune-config.<timestamp>.log` file. If necessary, make changes to the `amtune-env.properties` file based on the tuning recommendations.

5 **When you are satisfied with the tuning recommendations from running `amtune` in REVIEW mode, set `AMTUNE_MODE` to `CHANGE` in the `amtune-env.properties` file.**

6 **Check the debug log file for the results.**

In CHANGE mode, the `amtune` tool might need to restart the web container and OpenSSO Enterprise. When the operating system kernel parameters are changed, the `amtune` tool will recommend a system restart.

## ▼ To Tune a Remote Sun Directory Server

1 **FTP or copy `amtune.zip` to the remote Sun Directory Server host.**

In the change mode section of `amtune-env.properties` file, if `AMTUNE_TUNE_DS=true`, then the `amtune.zip` file is automatically generated.

2 **Unzip `amtune.zip`.**

3 **Set values for TOOL_HOME and JAVA_HOME.**

| | |
|---|---|
| Solaris, Linux, and AIX | `<TOOLS_DIR>/bin/unix/amtune` |
| Windows | `<TOOLS_DIR>\bin\windows\amtune.bat` |

4 **Edit the `amtune-env.properties` file to include Directory Server information.**

5 **Run the `amtune` tool with a password file for the Sun Directory Server Directory Manager.**

The Directory Server Directory Manager password must be inaccessible to non-owners and only readable by its owner . For example, you can run change the permissions mode of the password file by running the following command:

```
chmod 400
```

| | |
|---|---|
| UNIX or Linux | `amtune` |
| Windows | `amtune.bat`<br>On Windows, you must also execute `amtune.bat` with a password file. But `amtune.bat` does not check its file permission on Windows. |

In CHANGE mode, if `AMTUNE_TUNE_DS=true`, then the `amtune` tool will restart the Sun Directory Server instance.

If AMTUNE_TUNE_OS=true, then the amtune tool will tune the operating system kernel parameters and will recommend a system restart.

CHAPTER 3

# 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 OpenSSO Enterprise performance issues. Topics in this chapter include:

- "Tuning the LDAP Connection Pool and LDAP Configurations" on page 29
- "Tuning the Policy Cache" on page 31
- "Resolving Memory Issues" on page 33
- "More Resources" on page 36

## Tuning the LDAP Connection Pool and LDAP Configurations

The amtune tool provided by OpenSSO Enterprise tunes parameter values for the following three LDAP connection pools:

- Realm User Authentication LDAP Connection Pool
- Realm Data Store LDAP Connection Pool
- OpenSSO Enterprise Configuration Store and SMS LDAP Connection Pools

In deployments with a subrealm, you must also tune the subrealm 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 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 30
- "To Tune the Data Store LDAP Configuration" on page 30
- "To Tune the LDAP Configuration for the OpenSSO Enterprise Configuration Date Store" on page 30

## 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 OpenSSO Enterprise 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. If the Data Store Authentication module is used for authentication, then the recommended Data Store LDAP configuration settings are `MIN=8` and `MAX=64`. You can modify the settings under Console > Access Control > Realm > Data Store.

## To Tune the LDAP Configuration for the OpenSSO Enterprise Configuration Date Store

The configuration data store is used for storing all the OpenSSO Enterprise configurations and Policy Service configurations. Configuration data is stored in the `config` directory. The OpenSSO Enterprise server supports Sun Directory Server and the embedded OpenDS as the `config` data stores. You can configure the LDAP configuration for the `config` data store through the OpenSSO Enterprise administration console. Go to Configuration >Servers and Sites > server >Directory configuration.

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 OpenSSO Enterprise logging 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 OpenSSO Enterprise console. See the "Logging" section in Chapter 7, "Configuration Attributes," in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

# Tuning the Policy Cache

Two modes exist for client-side policy configuration: subtree mode and self mode. Based on the client configuration, server-side policy evaluation is done differently.

In subtree mode, all the policies from the root resource are evaluated. The high performance cost of evaluating high number of policies makes caching necessary. In self mode, only one resource is evaluated. Self mode is fast, and no caching is required. So there is no need to tune the policy cache when all the clients are running in self mode.

## Policy cache behavior

The policy cache is a two-level nested cache, with one hash map contained inside the other. The top level cache is the resource cache. The session cache is a second hash map inside the resource cache.

| | |
|---|---|
| Policy Resource cache | A hash map whose key is resource/rule name and the value is hash map of policy session cache. |
| Policy Session cache | A hash map whose key is `sessionid` and the value is map of policy decision objects. For each new resource a new hash map of session cache is created and stored in the policy resource cache. |

## Configuring the Policy Cache Limit

You can configure the policy cache by setting properties for both server and client.

### Configuring Server-Side Properties

The following two properties do not exist in the OpenSSO Enterprise administration console by default. These properties must be added manually in the advanced properties section of the OpenSSO Enterprise administration console:

`com.sun.identity.policy.resultsCacheResourceCap`

The default value is 100. This means that a maximum of 100 rules can be cached in subtree mode.

This property should be always equal to the total number of rules configured in the system. Otherwise, when the maximum cache limits are reached for the resource cache, and if a new rule or resource is accessed, then the oldest cached rule and all the sessions cached for that rule will be removed. If you have large number of rules, configure this value to the total number of most frequently accessed rules.

`com.sun.identity.policy.resultsCacheSessionCap`
The default value is 1000. Total number of policy objects is (100 *1000) or 100,000 maximum.

The `resourceCap` should be always tuned. The `SessionCap` should be tuned accordingly only when you observer high latency for policy requests or responses, and you observe repeated policy requests from the same policy agent for the same user. This usually does not occur unless the user session stays active for a very long period. The policies are also cached on the policy agent.

If you increase the `ResourceCap` value correspondingly, you should also reduce the `SessionCap` value to limit the total number of policy objects cached, and to maintain unchanged the maximum number of sessions supported on the server. The following table illustrates how the policy cache configuration effects the number of sessions supported. The SDK cache size is set to 10,000 for all of the tests. If the SDK cache is increased, the maximum number of sessions will be reduced accordingly.

**TABLE 3–1** Policy Session Cache Configuration and Number of Sessions Supported

| Policy Session Cache Configuration | Maximum Number of Sessions Supported |
|---|---|
| 1000<br>(100 * 1000 = 100,000 policy decision objects) | 200,000 |
| 2000<br>(100 * 2000 = 200,000 policy decision objects) | 150,000 |
| 3000<br>(100 * 3000 = 300,000 policy decision objects) | 90,000 |
| 4000<br>(100 * 4000 = 400,000 policy decision objects) | 40,000 |

## Configuring Client-Side Properties

The client-side SDK and policy agent cache properties apply only to Java EE policy agents. The properties do not apply to web agents.

`com.sun.identity.policy.client.resultsCacheResourceCap`
The default value 20. This means the Java EE policy agent can cache a maximum of 20 rules or resources.

This property should be set equal to the number of rules configured on the server for the FQDN the Agent is protecting. Otherwise, when the maximum cache limits are reached for the resource cache, and if a new rule or resource is accessed, then the oldest cached rule and all the sessions cached for that rule will be removed.

`com.sun.identity.policy.client.resultsCacheSessionCap`

The default value is 10000. This means the Java EE policy agent can cache a maximum of 10000 sessions per rule or resource. This property should be reduced or increased based on the memory available on the container.

The `ResourceCap` value should be always tuned. Since the policy agents co-exist with the application, you should increase or reduce the `SessionCap` on the policy agent based on the memory use of the application protected by the policy agent. You can increase the `SessionCap` value until you no longer observe frequent full GCs.

## Resolving Memory Issues

The `amtune` tool 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, use the OpenSSO Enterprise administration console to tune the following parameters:

- User cache/SDK cache

  Go to Configuration > Servers and Sites > server > SDK > SDK Caching Max Size
- Max Active Session the system should allow

  Go to Configuration > Servers and Sites > server > Maximum Sessions.
- Session Notification Thread Pool Size (Number of threads to process session notifications)

  Go to Configuration >Servers and Sites > server >Notification Pool Size.
- Session Notification Queue size

  Go to Configuration > Servers and Sites > server > Notification Thread Pool Threshold.
- Session Purge Delay (Number of minutes to delay the purge timed-out session)

  Go to Configuration > Servers and Sites > server > Session Purge Delay.

To tune the policy cache, see "Tuning the Policy Cache" on page 31.

Tuning Maximum Sessions

The tuning of this property depends on the JVM heap size configured in the web container where OpenSSO Enterprise is deployed. The minimum required JVM heap size for OpenSSO Enterprise is

| | |
|---|---|
| | 1024 MB, and the number of sessions supported for 1024 MB is approximately 7000. see the table below for various JVM heap sizes with the default configuration. |
| Tuning SDK Cache Max Size | The default value is set to 10000, This is suitable for most deployments. The SDK cache value can be increased to equal to the maximum number of sessions as long as you don't encounter frequent full GCs. Increasing this value results in slightly better performance, but will reduce the maximum number of sessions. |
| Tuning Session Notification Queue Size | The Notification Queue size should be less than or equal to 30% of the Max Sessions, up to a maximum value of 30,000. |

The following table lists the maximum number of sessions supported for various JVM heap sizes with the default tuning.

**TABLE 3–2** Maximum Number of Sessions Supported for Various JVM Heap Sizes

| JVM Heap Size | Max # of session supported |
|---|---|
| 3136 MB | 200,000 |
| 2560 MB | 145,000 |
| 1536 MB | 45,000 |
| 1024 MB | 7,000 |

These 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 64-bit JRE.
- 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 35 for 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. For CMT T1, T2, and T2 plus machines, use the formula: (number of cores) x 3 = threadpool size. The `amtune` tool sets this value based on the above rules, when `purgedelay` is set to 0, which is the default setting.
- 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 . For CMT T1, T2, and T2 plus machines, use the formula: (number of cores) x 4 = threadpool size. The notification threadpool size should be set manually by a multiple of 4 times the number of CPUs or cores. With this setting, if you still see problems such as frequent "Cannot send notification" or "Notification task queue full" errors in the `amSession` debug file, this indicates 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` tool 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 OpenSSO Enterprise 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 3136 MB. 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.

## 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://www.redbooks.ibm.com/abstracts/redp4285.html
- Java 5.0 Troubleshooting and Diagnostic Guide

  http://java.sun.com/j2se/1.5/pdf/jdk50_ts_guide.pdf
- Troubleshooting JRE 6 Deployment

  http://java.sun.com/developer/technicalArticles/javase/troubleshoot/

◆ ◆ ◆ APPENDIX A

# amtune-env.properties Reference

This appendix lists properties in the amtune-env.properties you must modify and verify before running the amtune tool. After you have modified the file to suit your deployment, you can run the amtune tool.

See "Using the amtune Tool" on page 25for usage details.

- "Tuning Modes" on page 37
- "Log Level Options" on page 39
- "Components to be Tuned" on page 39
- "Sun Web Server Settings" on page 42
- "Sun Application Server 9.1 and GlassFish v2 Settings" on page 44
- "OpenSSO Enterprise Settings" on page 47
- "Sun Directory Server Settings" on page 48
- "Special Performance Settings" on page 50

## Tuning Modes

### AMTUNE_MODE

| | |
|---|---|
| Description: | Based on this setting, the amtune tool will behave differently. |
| Required: | Yes |
| Sample Values: | |

| | |
|---|---|
| REVIEW | Suggests tuning recommendations only. In this mode, the amtune tool suggests tuning recommendations, but will not make any changes to the deployment environment. |

CHANGE Implements tuning recommendations. In this mode, `amtune` implements all of the tuning recommendations that you have defined in here, except for Sun Directory Server. See the note below for Sun Directory Server tuning.

Default Value: None

Additional Information: Use extreme caution while using CHANGE mode. In CHANGE mode, the `amtune` tool may restart the web container on which OpenSSO is deployed. The `amtune` tool may also recommend a system restart when the operating systems kernel parameters are changed.

For Operating System kernel and TCP parameter tuning, the `amtune` tool tunes the operating system parameters only on Solaris and Linux. The `amtune` tool does not tune the operating system parameters on AIX, Windows, MacOS or BSD variants.

Sun Directory Server tuning requires extra levels of confirmation. The `amtune` tool assumes that OpenSSO Enterprise will use an existing Sun Directory Server in non-exclusive mode, although other applications may use Directory Server. If the Directory Server is installed on a remote machine, it will not be tuned automatically. If the `amtune` tool detects that the Directory server is installed on a remote machine, it creates an `amtune.zip` file for tuning the remote Directory Server. For more information, see "To Tune a Remote Sun Directory Server" on page 27 in this document.

To selectively tune various components, see the section "AMTUNE_TUNE_*" on page 39 section of this document.

On Windows, use a forward slash ( `/` ) for file separators. Example: `c:/sun/webserver7`

For tuning multiple data stores, execute `amtune` multiple times using different values for `DS_*` parameters and `DIRMGR_PASSWORD`.

# Log Level Options

## AMTUNE_LOG_LEVEL

Description: Controls the logging of configuration data (calculated tuning values).

Required: Yes

Sample Values:

| | |
|---|---|
| TERM | The output is only displayed on the terminal. |
| FILE | The output is displayed on both terminal and in `amtune-config.<time stamp>.log` file. |

Default Value: None

Additional Information: Check the `<TOOLS_DIR>/<OPENSSO_URI>/logs/amtune-config.<time stamp>.log` file for the tuning parameters and their recommended values from each run.

# Components to be Tuned

## AMTUNE_TUNE_*

Description: Specifies components to be tuned by `amtune`.

These settings work in conjunction with the `AMTUNE_MODE` parameter setting. You can review or change recommended tunings of any set of these components.

Required: Yes

Properties Details

`AMTUNE_TUNE_OS=`
Only Solaris and Linux kernel and TCP parameters are supported for tuning.

`AMTUNE_TUNE_DS=`
Only Sun Directory Server is supported for tuning.

`AMTUNE_TUNE_WEB_CONTAINER=`
Only Sun Application Server 9.1, GlassFish v2, or Sun Web Server 7 are supported for tuning.

`AMTUNE_TUNE_OPENSSO=`
OpenSSO tuning.

Other Containers
$WEB_CONTAINER and $CONTAINER_INSTANCE_DIR values do not have to be filled in.

Sample Values: True or False

Default Value: None

Additional Information: Even if only `AMTUNE_TUNE_OPENSSO` is set to `true`, if Web Server 7.0 or Application Server 9.1 is the web container for OpenSSO, you must specify values for the following:

- `$WEB_CONTAINER`
- `$CONTAINER_ INSTANCE_DIR`
- `$WSADMIN_*` or `$ASADMIN_*`

The `amtune` tool determines whether these containers are running in 32- or 64-bit JRE mode. The `amtune` tool restarts the server to check its JRE mode and to determine how much heap size is available for setting OpenSSO cache and session entries. For other web containers, the `amtune` tool supports only 32-bit JRE. For other web containers, set `$AMTUNE_TUNE_WEB_CONTAINER` to `false`. Also note the following:

- `$WEB_CONTAINER` must be set to `other`.
- `$CONTAINER_INSTANCE_DIR` should be left blank.
- OpenSSO parameters will be tuned if the value for `AMTUNE_TUNE_OPENSSO` is set to `true`.

By default, the `amtune` tool runs based on the assumption that the following amount of memory (megabytes) is available for tuning OpenSSO when the web container (both Sun and non-Sun) is running with 32-bit JRE:

- (Sparc/x86/AIX) `AMTUNE_MAX_MEMORY_TO_USE_IN_MB_SOLARIS=3584`
- (Linux) `AMTUNE_MAX_MEMORY_TO_USE_IN_MB_X86=2341`
- (Windows)`AMTUNE_MAX_MEMORY_TO_USE_IN_MB_DEFAULT=1536`

The `amtune` tool also tunes OpenSSO Enterprise when it is deployed on WebSphere 6.1 and 7, and on AIX, although it does not tune AIX system parameters or WebSphere container parameters.

## Web Container Options

- "WEB_CONTAINER " on page 41
- "CONTAINER_INSTANCE_DIR" on page 41

### WEB_CONTAINER

Description: Specifies OpenSSO web container name and version.

Required: Yes

Sample Values:

| | |
|---|---|
| Sun Web Server 7 | `WS7` |
| Sun Application Server 9.1 or GlassFish v2 | `AS91` |
| Other Web Containers | `other` |

Default Value: None

Additional Information: For Web Server 7 and Application Server 9.1, the `amtune` tool tunes JRE heap and per-thread stack sizes, JVM garbage collection algorithms, container worker or acceptor thread, and queue sizes.

For other web containers, the `amtune` tool does not change JVM or container-specific parameters.

`$WEB_CONTAINER` must be set to `other`. It is impossible to detect whether a null value is mistakenly set for Web Server 7 or Application Server 9.1, or is intentionally set for other web containers.

### CONTAINER_INSTANCE_DIR

Description: Specifies the OpenSSO web container instance directory.

Required: If the you are using Sun Web Server 7 or Sun application Server 9.1, then this parameter is required.

Sample Values:

| | |
|---|---|
| Sun Web Server 7 | `/sun/webserver7/https-localhost` |
| Sun Application Server 9.1 | `/sun/appserver/domains/domain1` |

Default Value: None

Additional Information: If you have installed Sun Web Server or Sun Application Server in a non-default location, then change this value before running `amtune`.

On Windows, if a directory name has spaces, then use a short form such as `E:/PROGRA~1/GLASSF~1`.

# Sun Web Server Settings

The following parameters are required for tuning JVM options and container parameters of Sun Web Server 7.0.

## WSADMIN_*

Set the following parameters when `$WEB_CONTAINER= WS7`.

- "WSADMIN_DIR" on page 42
- "WSADMIN_USER" on page 43
- "WSADMIN_HOST" on page 43
- "WSADMIN_PORT" on page 43
- "WSADMIN_SECURE" on page 43
- "WSADMIN_CONFIG" on page 44
- "WSADMIN_HTTPLISTENER" on page 44

### WSADMIN_DIR

Description: Specifies Sun Web Server 7 installation location.

Required: Yes, when `$WEB_CONTAINER=WS7`

Sample Values:

| | |
|---|---|
| Solaris | `/opt/SUNWwbsvr7/bin` |
| Linux | `/opt/sun/webserver7/bin` |
| Windows | `E:/Progra~1/webserver7/bin` |

Default Value: None

## WSADMIN_USER

| | |
|---|---|
| Description: | Specifies Sun Web Server administrator. |
| Required: | Yes |
| Sample Values: | `admin` ( Sun Web Server default) |
| Default Value: | None |

## WSADMIN_HOST

| | |
|---|---|
| Description: | Specifies Sun Web Server administrative host name. |
| Required: | Yes |
| Sample Values: | `localhost` |
| Default Value: | None |

## WSADMIN_PORT

| | |
|---|---|
| Description: | Specifies Sun Web Server 7 administration port. |
| Required: | Yes |
| Sample Values: | `8888`<br>`8989` (Sun Web Server default) |
| Default Value: | None |
| Additional Information: | If this port is a secure port, set the `$WSADMIN_SECURE` value to `--ssl=true`.<br>If this port is not a secure port, set the `$WSADMIN_SECURE` value to `--ssl=false`. |

## WSADMIN_SECURE

| | | |
|---|---|---|
| Description: | Flag to indicate whether or not `$WSADMIN_PORT` is in SSL mode. | |
| Required: | Yes | |
| Sample Values: | | |
| | If the port is a secure port | `--ssl=true` |
| | If the port is not a secure port | `--ssl=false` |
| Default Value: | None | |

### WSADMIN_CONFIG

| | |
|---|---|
| Description: | Specifies Sun Web Server instance name. |
| Required: | Yes |
| Sample Values: | `hostname.domain.com`<br>This sample value is the config-name for the default instance `https-hostname.domain.com` |
| Default Value: | None |
| Additional Information | If you have non-default config-name instances, for example `https-test1`, enter its config-name `test1` here. |

### WSADMIN_HTTPLISTENER

| | |
|---|---|
| Description: | Specifies HTTP listener name. |
| Required: | Yes |
| Sample Values: | `http-listener-1` (Sun Web Server default) |
| Default Value: | None |

## Sun Application Server 9.1 and GlassFish v2 Settings

These following parameters are required for tuning JVM options and container parameters of Sun Application Server 9.1 and GlassFish v2.

### ASADMIN_*

Set these parameters when `$WEB_CONTAINER= AS91`.

- "ASADMIN_DIR" on page 45
- "ASADMIN_USER" on page 45
- "ASADMIN_HOST" on page 45
- "ASADMIN_PORT" on page 45
- "ASADMIN_SECURE" on page 46
- "ASADMIN_TARGET" on page 46
- "ASADMIN_HTTPLISTENER" on page 46
- "AMTUNE_WEB_CONTAINER_JAVA_POLICY" on page 46

## ASADMIN_DIR

|  |  |
|---|---|
| Description: | Specifies Sun Application Server 9.1 or GlassFish v2 installation location. |
| Required: | Yes |
| Sample Values: |  |

| | |
|---|---|
| Solaris | /opt/SUNWappserver/bin |
| Linux | /opt/sun/appserver/bin |
| Windows | E:/Progra~1/glassfish-v2/bin |

|  |  |
|---|---|
| Default Value: | None |

## ASADMIN_USER

|  |  |
|---|---|
| Description: | Specifies Sun Application Server 9.1 or GlassFish v2 administrator. |
| Required: | Yes |
| Sample Values: | admin (Sun Application Server or GlassFish default) |
| Default Value: | None |

## ASADMIN_HOST

|  |  |
|---|---|
| Description: | Specifies Sun Application Server or GlassFish administrative host name. |
| Required: | Yes |
| Sample Values: | localhost |
| Default Value: | None |

## ASADMIN_PORT

|  |  |
|---|---|
| Description: | Specifies Sun Application Server or GlassFish administrative port. |
| Required: | Yes |
| Sample Values: | 4848<br>4849 (Sun Application Server or GlassFish default) |
| Default Value: | None |
| Additional Information: | If this port is a secure port, set $ASADMIN_SECURE value to --secure.<br>If this port is not a secure port, leave the $ASADMIN_SECURE value blank. |

## ASADMIN_SECURE

Description: Flag that indicates whether or not Sun Application Server or GlassFish is in SSL mode.

Required: Yes

Sample Values:

| | |
|---|---|
| If the port is a secure port | `--secure` (Application Server 9.1 or GlassFish v2 default) |
| If the port is not a secure port | Leave this value blank. |

Default Value: None

## ASADMIN_TARGET

Description: This value is usually set to server with the assumption that this Application Server 9.1 or GlassFish v2 installation is used exclusively for OpenSSO Enterprise

Required: Yes

Sample Values: `server` (Default in Application Server 9.1 or GlassFish v2)

Default Value: None

## ASADMIN_HTTPLISTENER

Description: Specifies Sun Application Server HTTP listener name.

Required: Yes

Sample Values: `http-listener-1` (Default in Sun Application Server 9.1 or GlassFish v2)

Default Value: None

## AMTUNE_WEB_CONTAINER_JAVA_POLICY

Description: Specifies whether Sun Application Server or GlassFish evaluates java security policies listed in the Application Server `server.policy` file.

Required: Yes

Sample Values: false (Application Server or GlassFish default)

Default Value: `false`

Additional Information: Do not modify this parameter setting unless it is a unique deployment requirement. Evaluating Java security policies can add a significant performance overhead.

# OpenSSO Enterprise Settings

## SSOADM_LOCATION

Description: Specifies the directory where the `ssoadm` command-line interface is located.

Required: Yes

Sample Values: `<TOOLS_DIR>/<OPENSSO_URI>/bin`

where `<TOOLS_DIR>` is the directory in which `amtune.zip` is unzipped, and `<OPENSSO_URI>` is the deployment URI of OpenSSO.

Default Value: `<TOOLS_DIR>/<OPENSSO_URI>/bin`

## OPENSSOADMIN_USER

Description: Specifies administrator of OpenSSO 8.x.

Required: Yes,

Sample Values: `amadmin` (Default in OpenSSO)

Default Value: None

## OPENSSOSERVER_URL

Description: Specifies OpenSSO URL.

Required: Yes

Sample Values: `http://<HOST_NAME>:<PORT>/<OPENSSO_URI>` (OpenSSO Enterprise default)

Default Value: None

### REALM_NAME

Description: Realm names for which user data store LDAP connection pool need to be modified.

Use the pipe ( | ) character as a delimiter for multiple realms.

Required: Yes

Sample Values:

Top_Level_Realm

```
/
```

Top_Level_Realm and its sub-realm, subrealm1

```
/|subrealm1
```

Top_Level_Realm and two sub-realms

```
/|subrealm1|subrealm2
```

Default Value: None

Additional Information: For all the data stores under each realm, minimum and maximum LDAP connection pool sizes will be tuned.

## Sun Directory Server Settings

The parameters in this section are for tuning a Sun Directory Server instance where a user management or service management and configuration data store is installed. When the Directory Server instance is on a remote computer system, after `amtune.zip` is copied over and unzipped, `amtune` validates parameter values only on that remote computer system.

- “DS_HOST” on page 48
- “DS_PORT” on page 49
- “ROOT_SUFFIX” on page 49
- “DS_INSTANCE_DIR” on page 49
- “DS_TOOLS_DIR ” on page 49
- “DS_VERSION ” on page 50
- “DIRMGR_BIND_DN ” on page 50

### DS_HOST

Description: Specifies Sun Directory Server fully qualified domain name (FQDN ).

Required: Yes

| | |
|---|---|
| Sample Values: | *hostname.domain.com* |
| Default Value: | None |
| | Enter the official host name; do not enter an alias. |

## DS_PORT

| | |
|---|---|
| Description: | Specifies Sun Directory Server port. |
| Required: | Yes |
| Default Value: | None |

## ROOT_SUFFIX

| | |
|---|---|
| Description: | Specifies the root suffix of the organization. |
| Required: | Yes |
| Default Value: | None |

## DS_INSTANCE_DIR

| | |
|---|---|
| Description: | Specifies Sun Directory Server instance location |
| Required: | Yes |
| Default Value: | None |
| Additional Information: | Use a forward slash (`/`) for file separators on Windows Systems. |

## DS_TOOLS_DIR

| | |
|---|---|
| Description: | Sun Directory Server `dsadm/dsconf` tools bin directory. |
| Required: | Yes |
| Default Value: | None |
| Additional Information: | Use a forward slash (`/`) for file separators on Windows Systems. |

## DS_VERSION

| | |
|---|---|
| Description: | Sun Directory Server version. |
| Required: | Yes |
| Sample Values: | 5.2 or 6.3 |
| Default Value: | None |
| Additional Information: | Sun Directory Server 6.2 is not supported for tuning due to its data corruption issues. |

## DIRMGR_BIND_DN

| | |
|---|---|
| Description: | Directory Manager `BIND DN` for `$DS_INSTANCE_DIR`. |
| Required: | Yes |
| Sample Values: | `cn=Directory Manager` (Directory Server default) |
| Default Value: | None |

# Special Performance Settings

The following parameters mainly are used internally by `amtune`.

**Caution –** Do not modify these parameters unless tests show significant improvement in performance.

- "AMTUNE_PCT_MEMORY_TO_USE " on page 50
- "AMTUNE_MEM_MAX_HEAP_SIZE_RATIO" on page 51
- "AMTUNE_PER_THREAD_STACK_SIZE" on page 52
- "AMTUNE_*_MEMORY_TO_USE_IN_MB_*" on page 52

## AMTUNE_PCT_MEMORY_TO_USE

| | |
|---|---|
| Description: | Specifies a percentage value how much of the machine's available memory will be used by OpenSSO Enterprise. |
| Required: | Yes |
| Sample Values: | 0 to 100 |
| Default Value: | 75 |

Additional Information: Do not modify this percentage unless tests show significant improvement in performance.

OpenSSO Enterprise currently recommends at least 1 GB of RAM in deployment. OpenSSO can use a maximum of 4GB for 32-bit JRE. This is the per-process address space limit for 32-bit applications.

When you set `AMTUNE_PCT_MEMORY_TO_USE` to 100, the maximum space allocated for OpenSSO is the lesser of 4GB and 100% of available RAM for 32-bit JRE.

When you set `AMTUNE_PCT_MEMORY_TO_USE` to 0, OpenSSO is configured to use 256MB RAM.

This value is the driving force in tuning OpenSSO. The following values are derived from this setting:

| | |
|---|---|
| JVM memory use | Heap and new generation sizes. |
| Thread pool sizes | Web Server thread pool and OpenSSO Enterprise authentication, user and service/configuration data store LDAP connection pools and session notification thread pool. |
| Session entries | Maximum number of session entries. |

For 64-bit JRE, the `amtune` tool limits the initial heap size (-Xms) to 12 GB for Web Server 7 and Application Server 9.1/Glassfish v2, although it can be increased manually to a bigger heap size, if the Solaris operating system has at least twice as much virtual memory (swap space) as the desired initial JRE heap size. There is no limit for the maximum heap size (-Xmx).

Using 64-bit JRE, the user session cache size and number of sessionsare calculated by the `amtune` tool, and can be many times of those calculated in case for 32-bit JRE, depending on the available memory. Be sure to review these numbers and determine whether or not they are appropriate .

## AMTUNE_MEM_MAX_HEAP_SIZE_RATIO

Description: These parameters are used to calculate the maximum and minimum heap sizes. Options include:

- `AMTUNE_MEM_MAX_HEAP_SIZE_RATIO`
- `AMTUNE_MEM_MIN_HEAP_SIZE_RATIO`

Required: Yes

Sample Values:

| | |
|---|---|
| Maximum heap size ratio | 7/8 |
| Minimum heap size ratio | 1/2 |

Additional Information: Do not modify these ratios unless tests show significant improvement in performance.

Web Server 7, Application Server 9.1 and GlassFish v2 use about 1/8 of the OpenSSO Enterprise JRE process heap size, leaving about 7/8 for OpenSSO Enterprise. You should change these ratios only for 64-bit JRE. For 32-bit JRE, keep the default values.

## AMTUNE_PER_THREAD_STACK_SIZE

Description: Specifies available stack space per thread in JVM. Per-thread stack size is used to tune various thread-related parameters in OpenSSO Enterprise and its web container. Options include:

- `AMTUNE_PER_THREAD_STACK_SIZE_IN_KB`
- `AMTUNE_PER_THREAD_STACK_SIZE_IN_KB_64_BIT`

Required: Yes

Sample Values:

| | |
|---|---|
| 32–bit JRE | 128KB |
| 64–bit JRE | 512KB |

Default Value: None

Additional Information: Do not modify these values.

## AMTUNE_*_MEMORY_TO_USE_IN_MB_*

Description: Maximum amount of memory that should not be exceeded for 32-bit JRE on different platforms. `AMTUNE_MAX_MEMORY_TO_USE_IN_MB_X86` is used to limit the maximum JRE heap size on Linux installed on x86 hardware due to limitations on how much JRE heap size can be allowed even with 32-bit JRE.

Options include:

- `AMTUNE_MIN_MEMORY_TO_USE_IN_MB`
- `AMTUNE_MAX_MEMORY_TO_USE_IN_MB_SOLARIS`
- `AMTUNE_MAX_MEMORY_TO_USE_IN_MB_X86`
- `AMTUNE_MAX_MEMORY_TO_USE_IN_MB_DEFAULT` (for Windows)

Required: Yes

Default Values `AMTUNE_MIN_MEMORY_TO_USE_IN_MB=512`

`AMTUNE_MAX_MEMORY_TO_USE_IN_MB_SOLARIS=3584` (Sparc/x86/AIX)

`AMTUNE_MAX_MEMORY_TO_USE_IN_MB_X86=2341` (Linux)

`AMTUNE_MAX_MEMORY_TO_USE_IN_MB_DEFAULT=1536` (Windows)

Additional Information: Do not modify these values. If the maximum values are changed to higher numbers, the web container will not start on these platforms due to a JRE crash.