Chapter 1. Tuning OpenAM

This chapter covers key OpenAM tunings to ensure smoothly performing access and federation management services, and to maximize throughput while minimizing response times.

Note

The recommendations provided here are guidelines for your testing rather than hard and fast rules for every situation. Said another way, the fact that a given setting is configurable implies that no one setting is right in all circumstances.

The extent to which performance tuning advice applies depends to a large extent on your requirements, on your workload, and on what resources you have available. Test suggestions before rolling them out into production.

As a rule of thumb, an OpenAM server in production with a 3 GB heap can handle 100,000 sessions. Although you might be tempted to use a larger heap with a 64-bit JVM, smaller heaps are easier to manage. Thus, rather than scaling single servers up to increase the total number of simultaneous sessions, consider scaling out by adding more servers instead. The suggestions that follow pertain to production servers.

1.1. OpenAM Server Settings

OpenAM has a number of settings that can be tuned to increase performance.

1.1.1. General Settings

The following general points apply.

- Set debug level to `error`.
- Disable session failover debugging.
- Set container-level logging to a low level such as `error` or `severe`.

1.1.2. LDAP Settings

Tune both your LDAP data stores and also your LDAP authentication modules.

To change LDAP data store settings, browse to Access Control > *`Realm Name`* > Data Stores > *`Data Store Name`* in the OpenAM console. Each data store has its own connection pool and therefore each data store needs its own tuning.

Table 1.1. LDAP Data Store Settings

Property	Default Value	Suggestions
LDAP Connection Pool Minimum Size	1	The minimum LDAP connection pool size; a good tuning value for this property is 10. (`sun-idrepo-ldapv3-config-connection_pool_min_size`)
LDAP Connection Pool Maximum Size	10	The maximum LDAP connection pool size; a high tuning value for this property is 65, though you might well be able to reduce this for your deployment. Ensure your LDAP server can cope with the maximum number of clients across all the OpenAM servers. (`sun-idrepo-ldapv3-config-connection_pool_max_size`)

To change connection pool settings for the LDAP authentication module, browse to Configuration > Authentication > Core in the OpenAM console.

Table 1.2. LDAP Authentication Module Setting

Property	Default Value	Suggestions
Default LDAP Connection Pool Size	1:10	The minimum and maximum LDAP connection pool used by the LDAP authentication module. This should be tuned to 10:65 for production. (`iplanet-am-auth-ldap-connection-pool-default-size`)

1.1.3. Notification Settings

OpenAM has two thread pools used to send notifications to clients. The Service Management Service thread pool can be tuned in OpenAM console under Configuration > Servers and Sites > Default Server Settings > SDK.

Table 1.3. SMS Notification Setting

Property	Default Value	Suggestions
Notification Pool Size	10	This is the size of the thread pool used to send notifications. In production this value should be fine unless lots of clients are registering for SMS notifications. (`com.sun.identity.sm.notification.-threadpool.size`)

The session service has its own thread pool to send notifications. This is configured under Configuration > Servers and Sites > Default Server Settings > Session.

Table 1.4. Session Service Notification Settings

Property	Default Value	Suggestions
Notification Pool Size	10	This is the size of the thread pool used to send notifications. In production this should be around 25-30. (`com.iplanet.am.notification.-threadpool.size`)
Notification Thread Pool Threshold	5000	This is the maximum number of notifications in the queue waiting to be sent. The default value should be fine in the majority of installations. (`com.iplanet.am.notification.-threadpool.threshold`)

1.1.4. Session Settings

The session service has additional properties to tune, which are configured under Configuration > Servers and Sites > Default Server Settings > Session.

Table 1.5. Session Settings

Property	Default Value	Suggestions
Maximum Sessions	5000	In production this value can safely be set into the 100,000s. The maximum session limit is really controlled by the maximum size of the JVM heap which must be tuned appropriately to match the expected number of concurrent sessions. (`com.iplanet.am.session.maxSessions`)
Sessions Purge Delay	0	This should be zero to ensure sessions are purged immediately. (`com.iplanet.am.session.purgedelay`)

1.2. Java Virtual Machine Settings

This section gives some initial guidance on configuring the JVM for running OpenAM. These settings provide a strong foundation to the JVM before a more detailed garbage collection tuning exercise, or as best practice configuration for production.

Table 1.6. Heap Size Settings

JVM Parameters	Suggested Value	Description
`-Xms` & `-Xmx`	At least 1024m (2048m with embedded OpenDJ), in production environments at least 2048m to 3072m. This setting depends on the available physical memory, and on whether a 32 or 64-bit JVM is used.	-
`-server`	-	Ensures the server JVM is used
`-XX:PermSize` & `-XX:MaxPermSize`	Set both to 256m	Controls the size of the permanent generation in the JVM

JVM Parameters	Suggested Value	Description
`-Dsun.net.client.-defaultReadTimeout`	60000	Controls the read timeout in the Java HTTP client implementation This applies only to the Sun/Oracle HotSpot JVM.
`-Dsun.net.client.-defaultConnectTimeout`	High setting: 30000 (30 seconds)	Controls the connect timeout in the Java HTTP client implementation When you have hundreds of incoming requests per second, reduce this value to avoid a huge connection queue. This applies only to the Sun/Oracle HotSpot JVM.

Table 1.7. Garbage Collection Settings

JVM Parameters	Suggested Value	Description
`-verbose:gc`	-	Verbose garbage collection reporting
`-Xloggc:`	`$CATALINA_HOME/logs/gc.log`	Location of the verbose garbage collection log file
`-XX:+PrintClassHistogram`	-	Prints a heap histogram when a SIGTERM signal is received by the JVM
`-XX:+PrintGCDetails`	-	Prints detailed information about garbage collection
`-XX:+PrintGCTimeStamps`	-	Prints detailed garbage collection timings

JVM Parameters	Suggested Value	Description
`-XX:+HeapDumpOnOutOfMemoryError`	-	Out of Memory errors generate a heap dump automatically
`-XX:HeapDumpPath`	`$CATALINA_HOME/logs/heapdump.hprof`	Location of the heap dump
`-XX:+UseConcMarkSweepGC`	-	Use the concurrent mark sweep garbage collector
`-XX:+UseCMSCompactAtFullCollection`	-	Aggressive compaction at full collection
`-XX:+CMSClassUnloadingEnabled`	-	Allow class unloading during CMS sweeps

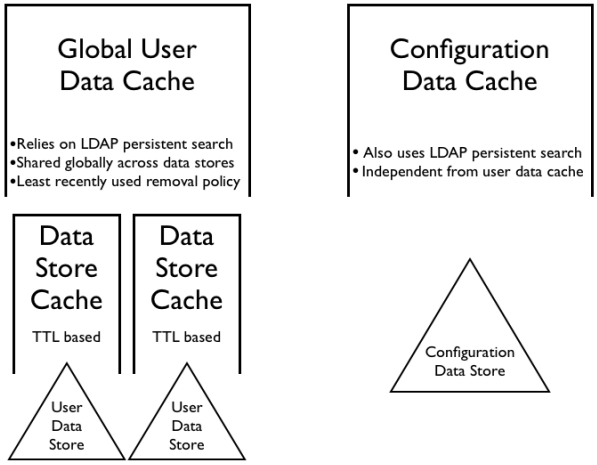
1.3. Caching in OpenAM

OpenAM caches data to avoid having to query user and configuration data stores each time it needs the information. By default, OpenAM makes use of LDAP persistent search to receive notification of changes to cached data. For this reason, caching works best when data are stored in a directory server that supports LDAP persistent search.

OpenAM has two kinds of cache on the server side that you can configure, one for configuration data and the other for user data. Generally use the default settings for configuration data cache. This section mainly covers the configuration choices you have for caching user data.

OpenAM server has two levels of user data caching. The global user data cache is dirtied by LDAP persistent search. When the global user data cache is enabled (as it is by default), the individual data store caches are not useful because all requests for data go through the global cache. The individual data store caches are therefore only useful when the global user data cache is disabled.

The following diagram depicts the two kinds of cache, and also the two levels of user data caching.



The rest of this section concerns mainly settings for global user data cache and for SDK clients. For a look at data store cache settings, see Table 1.1, “LDAP Data Store Settings”.

1.3.1. Overall Server Cache Settings

By default OpenAM has caching enabled both for configuration data and also for user data. This setting is governed by the server property `com.iplanet.am.sdk.caching.enabled`, which by default is `true`. When you set this advanced property to `false`, then you can enable caching independently for configuration data and for user data.

Procedure 1.1. To Turn Off Global User Data Caching

Disabling caching can have a severe negative impact on performance. This is because, when caching is disabled, OpenAM must query a data store each time it needs data.

If, however, you have at least one user data store that does not support LDAP persistent search, such as a relational database or an LDAP directory server that does not support persistent search, then you must disable the *global* cache for user data. Otherwise user data caches cannot stay in sync with changes to user data entries.

1. In the OpenAM console, browse to Configuration > Servers and Sites > *Server Name* > Advanced.

2. Set `com.iplanet.am.sdk.caching.enabled` to `false` to disable caching overall.

3. Set `com.sun.identity.sm.cache.enabled` to `true` to enable configuration data caching.

   All supported configuration data stores support LDAP persistent search, so it is safe to enable configuration data caching.

   You must explicitly set this property to `true`, because setting `com.iplanet.am.sdk.caching.enabled` to `false` in the previous step disables both user and configuration data caching.

4. Save your work.

Procedure 1.2. To Change the Maximum Size of Global User Data Cache

With a large user data store and active user base, the number of user entries in cache can grow large.

1. In the OpenAM console, browse to Configuration > Servers and Sites > Default Server Settings > SDK.

2. Change the value of SDK Caching Max. Size, and then Save your work.

   There is no corresponding setting for configuration data, as the number of configuration entries in a large deployment is not likely to grow nearly as large as the number of user entries.

1.3.2. Caching Properties For Java EE Policy Agents & SDK Clients

Policy agents and other OpenAM SDK clients can also cache user data, using most of the same properties as OpenAM server as described in Table 1.8, “OpenAM Cache Properties” . Clients however can receive updates by notification from OpenAM or, if notification fails, by polling OpenAM for changes.

Procedure 1.3. To Enable Notification & Polling For Client Cache Updates

This procedure describes how to enable change notification and polling for policy agent user data cache updates. When configuring a custom OpenAM SDK client using a .properties file, use the same properties as for the policy agent configuration.

1. In OpenAM console, browse to Access Control > *Realm Name* > Agents > *Agent Type* > *Agent Name* to view and edit the policy agent profile.

2. On the Global tab page, check that the Agent Notification URL is set.

   When notification is enabled, the agent registers a notification listener with OpenAM for this URL.

   The corresponding property is `com.sun.identity.client.notification.-url`.

3. For any changes you make, Save your work.

   You must restart the policy agent for the changes to take effect.

1.3.3. Cache Settings

The table below provides a quick reference, primarily for user data cache settings.

Notice that many properties for configuration data cache have `sm` (for Service Management) in their names, whereas those for user data have `idm` (for Identity Management) in their names.

Table 1.8. OpenAM Cache Properties

Property	Description	Default	Applies To
`com.iplanet.am.sdk.cache.-maxSize`	Maximum number of user entries cached	10000	Server & SDK
`com.iplanet.am.sdk.caching.-enabled`	Whether to enable caching for both configuration data and also for user data. If `true`, this setting overrides `com.sun.identity.-idm.cache.enabled` and `com.-sun.identity.sm.cache.enabled`. If `false`, you can enable caching independently for configuration data and for user data using the aforementioned properties.	`true`	Server & SDK
`com.iplanet.am.sdk.remote.-pollingTime`	How often in minutes the SDK client such as a policy agent should poll OpenAM for modified user data entries.	1 (minute)	SDK

Property	Description	Default	Applies To
	The SDK also uses this value to determine the age of the oldest changes requested. The oldest changes requested are 2 minutes older than this setting. In other words, by default the SDK polls for entries changed in the last 3 minutes. Set this to 0 or a negative integer to disable polling.		
`com.sun.am.event.-notification.expire.time`	How long OpenAM stores a given change to a cached entry, so that clients polling for changes do not miss the change.	30 (minutes)	Server only
`com.sun.identity.idm.cache.-enabled`	If `com.iplanet.am.sdk.caching.-enabled` is `true`, this property is ignored. Otherwise, set this to `true` to enable caching of user data.	`false`	Server & SDK
`com.sun.identity.idm.cache.-entry.default.expire.time`	How many minutes to store a user data entry in the global user data cache	30 (minutes)	Server & SDK
`com.sun.identity.idm.cache.-entry.expire.enabled`	Whether user data entries in the global user data cache should expire over time	`false`	Server & SDK
`com.sun.identity.idm.remote.-notification.enabled`	Whether the SDK client such as a policy agent should register a notification listener for user data changes with the OpenAM server. The SDK client uses the URL specified by `com.sun.identity.-client.notification.url` to register the listener so that OpenAM knows where to send notifications.	`true`	SDK

Property	Description	Default	Applies To
	If notifications cannot be enabled for some reason, then the SDK client falls back to polling for changes.		
`com.sun.identity.sm.cache.-enabled`	If `com.iplanet.am.sdk.caching.-enabled` is `true`, this property is ignored. Otherwise, set this to `true` to enable caching of configuration data. It is recommended that you always set this to `true`.	`false`	Server & SDK