Job Reference Guide

SLAMD Distributed Load Generation Engine

Version 1.8.2

June 2004

Contents

Introduction

As a load generation and performance analysis tool, SLAMD is extremely flexible. Although it was originally written for the purpose of benchmarking LDAP directory servers, great care has been taken to ensure that it can be used to measure the performance of virtually any network-based application. This has proven quite successful, as SLAMD has been used with Web and messaging servers, as well as to drive load against Web-based applications like the Sun ONE Identity, Portal, and Calendar servers.

One of the ways in which SLAMD maintains this flexibility is through the use of an API that can be used to develop new jobs. This makes it very easy to test new kinds of applications, or to test existing applications in different ways. This has been used extensively during internal testing, and the result is that a number of different kinds of jobs are provided with SLAMD in the default installation. This document describes many of the default jobs that are available and provides information on how they can be used to perform a variety of testing.

The Utility Jobs

The so-called "utility jobs" are not really jobs that are meant to do performance testing or load generation, but rather perform some useful function in order to facilitate testing using other jobs. SLAMD provides two such jobs: the null job and the exec job.

Null Job

As its name implies, the null job doesn't do anything. Or more accurately, the null job uses one or more clients to do nothing until it has run for the maximum duration, the stop time has been reached, or it is cancelled by an administrator.

Despite the fact that it doesn't do anything, the null job can actually be rather useful. Because it is technically a SLAMD job, you can make it dependent upon other jobs, and you can make jobs dependent upon it. Doing this allows you to ensure that there is a delay between two jobs without having to know when the first will end so that you can schedule the second accordingly.

For example, consider a case in which you wish to use SLAMD to perform two optimizing jobs (e.g., an LDAP SearchRate job followed by an LDAP AuthRate job). Because they are optimizing jobs, they will run for some number of iterations, but you do not know exactly how many. Without using the null job, you would have to take a guess at how many iterations the LDAP SearchRate job would require to find the optimal performance, then determine how long it would take to perform all those iterations, and finally schedule the LDAP AuthRate job with a start time far enough beyond that to allow for the desired pause between the jobs. This will work if your guess is accurate, but if you overestimated the length of time between jobs, it can mean that they systems are idle for longer than you wanted rather than actively processing other performance tests. If you underestimated the length of time required for the LDAP SearchRate optimizing job, then you will not have the desired delay between the jobs, and based on the way that the jobs have been configured you could even have some iterations of both jobs running at the same time.

The best way to avoid this problem is through the use of the null job. You should first schedule the LDAP SearchRate optimizing job as you normally would. Then, you should schedule a null job and in the advanced scheduling options you can specify that the null job should be dependent upon the LDAP SearchRate optimizing job (make sure that it is dependent upon the optimizing job and not simply one of the iterations of that optimizing job). Finally, schedule the LDAP AuthRate job, making it dependent upon the null job. For both the null job and the LDAP AuthRate optimizing job, you should use the default start time so that they will be eligible to start running as the job on which they are dependent has completed.

Because a null job does not actually do anything, there are no job-specific parameters that you will need to specify to alter the way that it operates.

Exec Job

The exec job will execute an arbitrary command on the client system(s) on which you have scheduled it to run. This is intended to be used to perform cleanup tasks or any other kinds of processing that you might need on the client system.

You can also use the exec job to perform cleanup operations on the server that you are testing. For example, if you are using the LDAP AddRate job to add entries to an LDAP directory server, you may wish to restore the original database before running the test again. To do this, you can start a SLAMD client on the server system and have that client run in restricted mode. If a client is running in restricted mode, then the server will only ask that client to process jobs for which that client was explicitly listed in a set of requested clients when the job was scheduled. See the SLAMD Administrator's Guide for more information on running clients in restricted mode.

When scheduling an instance of the exec job, the following parameters are available to be customized:

Command to Execute

This is used to specify the command that the client should execute. This may be a single command, or it may also include command-line arguments to provide to the command when it is executed. In order for it to complete successfully, you should ensure that the client can find the command you wish to run. That means that you should ensure that the command is either in the path, or you should provide the path in the command itself.

Working Directory

This is used to provide the working directory that should be used when running the specified command. This is an optional parameter, and if no value is specified, the current working directory of the SLAMD client application will be used.

Environment Variables

This is used to provide a set of environment variables that should be defined when running the specified command. This is optional, and if it is left blank, the SLAMD client's environment will be used. If any values are provided, then they should be given with one variable per line. Each environment variable definition should consist of the variable name followed by an equal sign and the value to use for the variable. There should not be any extra spaces around the equal sign.

Log Command Output

This indicates whether the output (both standard out and standard error) of the command will be captured and logged to the SLAMD server. If it is, then that output will be available in the "Messages Logged" section of the job execution data when the job is complete. It will also be written to the SLAMD log file if the appropriate log level (Job Processing) is enabled.

Throughput Test Job

The throughput test job provides a means of measuring the network throughput between a server and one or more other systems. It is intended to work in conjunction with the throughput test server, which is a utility provided with SLAMD in the **`tools`** directory under the server installation. Consult the **`tools/ThroughputTest/README.server`** file for information on starting the throughput test server.

Once the throughput test server is running the throughput test job can be used to interact with it. Performing this task as a SLAMD job rather than using the standalone throughput test client can provide a number of benefits, including the ability to easily use multiple threads on a single client system or use multiple client systems to generate network load concurrently, and the ability to graph the performance over time to determine whether it is consistent.

The throughput test job offers the following configuration parameters:

Server Address

This is used to specify the address of the throughput test server. Either an IP address or resolvable hostname may be provided. This is a required parameter.

Server Port

This is used to specify the port on which the throughput test server is listening. By default, the server listens on port 3333.

Read Buffer Size

This is used to specify the size in bytes of the buffer to use when reading data from the server. The default buffer size is 8192 bytes.

Megabytes to Transfer

This is used to specify the maximum amount of data in megabytes that each client thread should read from the server before calculating the total throughput. This is an optional argument, and if no value is provided (or a negative value is given), then the test will run until the stop time or maximum duration has been reached, or until the job has been cancelled by an administrator.

TCP Replay Job

The TCP replay job may be used to replay captured TCP traffic against a target server. The data to replay should be captured using the TCPCapture tool provided with SLAMD in the `tools/TCPReplay` directory, and can consist of basically any kind of TCP communication in which the interaction with the remote system occurs using a single port (more complex protocols like FTP that might use multiple ports to communicate are currently not supported). For more information on using the TCPCapture tool (or the associated TCPReplay command-line utility), see the README files in the `tools/TCPReplay` directory.

The following parameters are available for use with the TCP Replay job:

Target Server Address

This specifies the address of the server that should be targeted when replaying the captured data. This is a required parameter, but it may be either an IP address or resolvable name.

Target Server Port

This specifies the port of the server that should be targeted when replaying the captured data. This must be provided, and it must be an integer value between 1 and 65535, inclusive.

Capture File URL

This specifies the URL to the capture file containing the data to replay against the server. This URL may reference a capture file on a remote server (using a protocol like HTTP or FTP) or the local filesystem (using a FILE URL).

```
http://server.example.com/capture.data
https://server.example.com/capture.data
ftp://server.example.com/data/capture.data
http://user:password@server.example.com/capture.data
https://user:password@server.example.com/capture.data
ftp://user:password@server.example.com/data/capture.data
```

or a FILE URL that references a file on the local filesystem, like:

```
file:/data/capture.data
```

Preserve Original Timing

This indicates whether the client should attempt to preserve the original timing used by the client when the data was originally captured. If this option is selected, then the client will sleep for variable lengths of time between replaying each capture packet to simulate the delays that a real client would introduce when communicating with the server.

Timing Multiplier

This provides a multiplier that will be applied to the sleep time between each packet if the client is configured to preserve the original timing between requests. The default value is 1.0, which means that the client will attempt to send the packets at the same rate as the original client. Specifying a lower value will cause the client to try to send the packets at a faster rate than the original client (e.g., a value of 0.5 will cut the sleep time in half, which could effectively double the rate

at which the packets are sent), while a higher value will slow them down (e.g., a value of 3.0 will triple the sleep times between each packet).

Fixed Delay Between Packets

This specifies the delay in milliseconds between individual packets that should be used if the client should not attempt to preserve the original timing. A value of zero indicates that the client should not attempt to sleep at all between packets and therefore send the packets as fast as possible.

Maximum Number of Iterations

This specifies the maximum number of times that the client should replay the entire data set before stopping. A value of -1 indicates that there should not be any limit on the number of iterations, and that the client should continue running until it determines that it should stop for some other reason (e.g., the stop time arrives or the job is cancelled by an administrator).

Delay Between Iterations

This specifies the length of time in milliseconds that the client should sleep between iterations through the entire data set. A value of zero indicates that the client should not sleep at all between iterations through the data set.

Load Variance Test Job

The load variance test job can be used to test load variance definitions to determine whether they will theoretically result in the desired load pattern. This can be accomplished by viewing the graph resulting from processing on the job. It operates by incrementing a counter associated with each active thread at regular intervals so that the shape of the resulting graph will depict the change in load over time (assuming that the amount of load generated is only directly related to the number of active threads).

The following parameters are available for use with the load variance test job:

Load Definition URL

This specifies the URL to the file containing the load variance definition to use. The URL may reference a file on the local filesystem using the FILE protocol, or

a remote server using the HTTP, HTTPS, or FTP protocol. The following forms may be accepted:

```
file:/data/variance.definition
http://server.example.com/variance.definition
http://user:password@server.example.com/variance.definition
https://server.example.com/variance.definition
https://user:password@server.example.com/variance.definition
ftp://server.example.com/data/variance.definition
ftp://user:password@server.example.com/data/variance.definition
```

Loop Load Variance Definition

This checkbox indicates whether the job should repeat the load definition once the end is reached. By default, this is disabled, which means that once the client reaches the end of the variance definition, processing will continue with the same number of active clients that were running at the end of that variance definition. Note that if this option is enabled, then it is highly advised that the load definition either end with no active threads or that it begin by setting a fixed number of threads active. If this is not done, then the next pass through the variance definition will not start with the same number of active threads and therefore will likely not produce the intended result.

Update Interval (ms)

This specifies the length of time in milliseconds between updates of the counter used by each thread. Smaller values have the potential to result in a graph that more closely matches the theoretical shape of the graph, but values that are too small will result in too much thread contention and could become smaller than the resolution that can be achieved using the sleep timer. This value should not need to be changed in most cases.

The LDAP Search Jobs

In most LDAP directory servers, searches comprise a very large portion of the total set of operations performed. As such, when benchmarking a directory server, it is important to be able to accurately measure the performance of search operations. SLAMD provides a few jobs to help with this, and they are discussed in this section.

LDAP SearchRate

The LDAP SearchRate job provides a very useful means of measuring search performance in a directory server. It is an all-purpose tool that can be used in a wide variety of scenarios. In many ways, it is similar to the **`searchrate`** command line utility provided in the Sun ONE Directory Server Resource kit, but it does have additional functionality, including the ability to communicate over SSL and the ability to target a particular search rate by including a delay between requests.

The parameters that are available when scheduling an LDAP SearchRate job are as follows:

Directory Server Host

This specifies the address of the directory server system that the clients should access when searching. It may be either an IP address or a fully-qualified name as long as it is resolvable to clients. It may also be the address of a load balancer or directory proxy server, as long as clients can send an LDAP request to it and receive the appropriate responses.

Directory Server Port

This specifies the port number of the directory server system that the clients should access when searching. If SSL is to be used, then this should be the secure port number for the server.

Bind DN

This specifies the DN of the user as whom to bind when performing the searches. If no value is provided, then the searches will be performed anonymously.

Bind Password

This specifies the password for the bind DN that has been provided. If no password is provided, then the searches will be performed anonymously.

Search Base

This specifies the DN of the entry to use as the search base. This entry must exist in the server in order for the searches to be successful. If no DN is provided, the root DSE will be used as the search base.

Note that the search base can actually specify a range of values by including a numeric range in brackets in this DN. For example, a search base of:

```
uid=user.[1-1000000],ou=People,dc=example,dc=com
```

will cause a random number to be chosen between 1 and 1000000 (inclusive), and the number chosen will be used in place of the bracketed range. Also, if a colon is used instead of a dash to separate the numbers in the numeric range, like:

```
uid=user.[1:1000000],ou=People,dc=example,dc=com
```

then a number will not be chosen at random, but rather the values will be chosen sequentially. That is, in the above case the first search base used will have a value of 1, the second a value of 2, the third a value of 3, and so on. Once all values in the range have been exhausted, then it will loop back around to the first value. Note that the sequential ordering can only be maintained across a single client, although it can be maintained across multiple threads on that client. Therefore, if multiple clients are to be used, then each value will be used once per client.

Search Scope

This specifies the scope to use when performing the searches. The value may be "Search Base Only", "One Level Below Base", or "Whole Subtree".

Search Filter

This specifies the search filter to use when performing the searches. This can be a single search filter, but it can also include a bracketed numeric range to make it possible to specify a pattern to use when generating the search filter. For example, the filter:

```
(uid=user.[1-1000000])
```

will cause a number between 1 and 1000000 (inclusive) to be chosen and used in place of the bracketed range. Similarly, using a colon instead of a dash, like:

```
(uid=user.[1:1000000])
```

will cause the numbers to be chosen sequentially, starting at the lower value and increasing until the largest value is used, at which time it will start over at the lowest again. Note that the sequential ordering can only be maintained across a single client, although it can be maintained across multiple threads on that client. Therefore, if multiple clients are to be used, then each value will be used once per client.

Filter File URL

This specifies the URL to a file containing the filters to use when performing the searches. This is an alternative to specifying a single search filter or filter pattern, but if you do use a filter file, then you should not use a single search filter or filter pattern.

The value of this parameter should be a URL. It may be a URL that references a file on a remote server, like:

```
http://server.example.com/filters.txt
https://server.example.com/filters.txt
ftp://server.example.com/data/filters.txt
http://user:password@server.example.com/filters.txt
https://user:password@server.example.com/filters.txt
ftp://user:password@server.example.com/data/filters.txt
```

or a FILE URL that references a file on the local filesystem, like:

```
file:/data/filters.txt
```

If a file is provided, then it should contain one search filter per line. For each search performed, the client will choose a line at random and use the filter contained on that line. Filters provided in a filter file may not contain a bracketed numeric range like may be used with the single search filter.

Note that using a very large filter file has been seen to degrade search performance because it slows the client's ability to issue searches to the server. If this occurs, and if you have multiple clients, then you may wish to split the large filter file into multiple smaller files and use the GetFile servlet provided with SLAMD to distribute a different file to each of the clients. See the SLAMD Administrator's Guide for more information on the GetFile servlet.

Attributes to Return

This specifies the set of attributes that the clients should request when performing the searches. If no attribute list is provided, then the server should return all non-operational attributes that the authenticated user has permission to access. If multiple attributes are to be requested, then they should be included with one attribute per line. If you do not want the server to return any attributes, then you can simply provide a value of "`1.1`" as defined in RFC 2251, section 4.5.1.

Warm Up Time

This specifies the length of time in seconds that the job should be allowed to run before it will start collecting statistics. Using a warm up time makes it possible to ensure that all clients and threads have started running and there is a consistent load on the directory server before the job starts collecting statistics. Therefore, it is recommended in most cases that a small warm up time be allowed.

Note that the warm up time is considered part of the overall duration of the job. Therefore, if you want to collect statistics for a fixed length of time but also want to use a warm up time, then the job duration should be increased to account for the warm up time.

Cool Down Time

This specifies the length of time before the job ends that it should stop collecting statistics. Using a cool down time helps ensure that all clients and threads are still active and there is a consistent load on the directory server when statistics collection has ended. Therefore, it is recommended in most cases that a small cool down time be allowed.

Note, however, that a cool down time will only be used if the job ends because it ran for the maximum duration or the stop time has been reached. If the job ends

because it performed the maximum number of iterations, because it was cancelled by an administrator, or for any other reason, then the client will not be able to determine when the job will end and therefore will not be able to use the cool down time.

It should also be noted that, like the warm up time, the cool down time is considered part of the overall job duration. Therefore, if you are using a cool down time you may wish to increase the job duration so that the desired length of time is actually spent collecting statistics.

Search Size Limit

This specifies the maximum number of entries that the server should return from a single search operation. If the search criteria provided by the client matches more than this number of entries, then the search will end after this number of entries have been returned to the client, and the client will proceed with the next search request.

Search Time Limit

This specifies the maximum length of time in seconds that should be spent processing any single search request. This time limit actually has two purposes. First, it will be included in the search request sent to the server so that the server knows it should not spend longer than this amount of time processing the request. In addition, it will be used on the client side so that it will also stop processing the request after this length of time, and the client will abandon the current request and continue on with the next.

Note that there has been a bug observed in the LDAP SDK for Java that is used for the underlying LDAP communication between the SLAMD client and the directory server. With this bug, it has been seen in some rare cases a client thread can actually become hung while waiting for results from a search operation. In such cases, specifying a search time limit has allowed the client to recover from that situation once this length of time has passed. Therefore, it is recommended that you do provide a time limit when scheduling an LDAP SearchRate job as a workaround to this problem, with this time limit set longer than you believe will be required for any legitimate search operation to complete.

Time Between Requests

This specifies the length of time in milliseconds that should be allowed between search requests sent to the server. Note that this length of time is between the start of one request and the start of the next, rather than between the end of one and the beginning of the next. If any single search request takes longer than this

length of time to complete, then there will be no delay before starting the next search.

This feature it possible to target a specific number of search operations per second by adjusting the number of clients, threads per client, and delay between requests accordingly. The targeted number of searches per second can be determined using the following formula:

$$o = 1000 / d * c * t$$

where o is the number of operations per second, c is the number of clients, t is the number of threads per client, and d is the delay between requests. If you know the number of available clients, threads per client, and number of operations per second you wish to target, you can determine the delay to use with a slight variation of this formula:

$$d = 1000 * c * t / o$$

Use SSL

This indicates whether the client should use SSL when communicating with the directory server. If SSL is to be used, then the provided port number should be the secure port of the directory server, and you should either choose to blindly trust any certificate or provide the appropriate key and trust store information.

Blindly Trust Any Certificate

This indicates whether the client should blindly trust any SSL certificate presented by the directory server when communicating over SSL. This is very useful for testing because it does not require that all the clients be given information necessary to trust the certificate presented by the server, however it is not recommended for use in a production environment because of the weaker security it provides.

SSL Key Store

This specifies the location of the JSSE key store that should be used when communicating with the directory server over SSL. This should only be needed if the server is configured to require SSL client authentication. Note, however, that this client does not support the SASL EXTERNAL authentication mechanism, so even if a client is required to present a certificate to the directory server , that client certificate will not be used to perform LDAP authentication.

SSL Key Store Password

This specifies the password that should be used when accessing information in the JSSE key store. This should not be necessary unless a JSSE key store is required.

SSL Trust Store

This specifies the location of the JSSE trust store that should be used when communicating with the directory server over SSL and the client has not been configured to blindly trust any certificate presented by the server.

SSL Trust Store Password

This specifies the password that should be used when accessing information in the JSSE trust store. This should not be necessary unless a JSSE trust store is required, and then only if the trust store is in a format that requires a password to access the information contained in it.

Number of Iterations

This specifies the number of search requests that should be sent by each client thread. Once all client threads have completed this number of iterations, then the job will be complete. A value of -1 indicates that there should be no limit to the total number of iterations to be performed.

If a number of iterations is specified, then the total number of requests that will be sent to the server is equal to the product of the number of iterations, the number of clients, and the number of threads per client. Note, however, that this specifies the maximum number of iterations to be performed per thread. It is possible that fewer searches will be performed if the job stops early for some other reason (e.g., maximum duration, stop time, or cancelled by an administrator).

Always Disconnect

This indicates whether the client should use a separate connection for each search operation. By default, each client thread will establish one connection at the beginning of the job and use it for all operations performed by that thread. If you check this box, then each operation will consist of establishing a connection, binding to the server (if a bind DN and password were provided), performing the search, and disconnecting.

Follow Referrals

This indicates whether the client should follow any referrals that it may encounter when searching. By default, no referrals will be followed. Checking this checkbox will cause the client to follow all referrals.

LDAP Weighted SearchRate

The LDAP Weighted SearchRate job is very similar to the LDAP SearchRate job, but the main advantage that it provides is the ability to use two different filter patterns and specify a weight that indicates how frequently each pattern is used. This makes it possible to perform searches in accordance with the 80/20 rule, in which case 80% of the searches are targeted at 20% of the data. For example, if you have 1000000 user entries in the directory server and they all have user IDs that look like "user.#", where # is a number between 1 and 1000000, then this job will allow you to target them with two filters like:

```
(uid=user.[1-200000])
(uid=user.[200001-1000000])
```

in which case you would specify that the first filter should be used approximately 80% of the time.

Not only does this job help you more accurately simulate access patterns that will occur in your production directory, but it can also be very useful for systems in which there is not enough memory to hold the entire database in cache. As long as you can ensure that at least the most frequently accessed portion of the data is in cache, you should be able to achieve notably better performance with a weighted access pattern than with one that is completely random.

Because the LDAP Weighted SearchRate job is so similar to the LDAP SearchRate job, the majority of the configurable parameters are the same as well. Because of that, only the differences between the parameters available for the two jobs are discussed here. Those differences are:

- In the Weighted LDAP SearchRate job, the search base specified must be a single DN. It may not contain a bracketed range of numeric values.
- In the Weighted LDAP SearchRate job, it is not possible to provide a file containing a list of search filters. Therefore, only search filters or filter patterns may be used. Those filters should be provided in the "**Search Filter 1**" and "**Search Filter 2**" parameters.

- The Weighted LDAP SearchRate job introduces a new configuration parameter, "**Filter 1 Percentage**", which specifies the frequency with which the filters should be used. The value of this parameter should be an integer between 0 and 100 (inclusive), and it specifies the portion of the time that the filter or filter pattern specified in the "**Search Filter 1**" parameter will be used. The filter or filter pattern specified in the "**Search Filter 2**" parameter will be used (100 - filter 1 percentage) percent of the time.

LDAP Prime Job

The LDAP Prime Job provides a fast means of priming the caches in a directory server. This is important because the caches should be full in order to ensure that you can obtain consistent performance results. It is true that you can use a subtree "`(objectClass=*)`" search to load all the entries into the entry cache, but there are a number of problems with that approach:

- It can take a long time to complete. It can only use a single client, and only one thread on that client.
- It does not do a very good job of priming the database cache. It only loads information from the id2entry database file, but does not bring in data from any of the other indexes.
- It does not provide a means for selectively loading a portion of the entries in the directory. This can be useful if the database is too large to fit in memory.

The LDAP Prime Job provides a solution for these problems and provides a more effective means of priming the directory server. In order to do this, it requires that the data set contains an attribute, indexed for equality, whose value is a sequentially incrementing integer. By giving the LDAP Prime Job an upper and lower bound, it will divide that range of values into smaller portions and give each client a different portion of the range of values. Each client will then divide its subset and give a different range to each of its threads. Each client thread will then perform individual searches to retrieve the entries with each of those values.

The parameters that may be specified when scheduling an LDAP Prime Job are as follows:

Directory Server Host

This specifies the address of the directory server that is to be primed. It may be either an IP address or a resolvable fully-qualified hostname. It should not be the address of a load balancer or directory proxy server because balancing the load across multiple servers would prevent any server from being primed properly.

Directory Server Port

This specifies the port number of the directory server that is to be primed. The LDAP Prime Job does not provide the option to communicate over SSL, and therefore this should be the non-secure port of the server.

Bind DN

This specifies the DN of the user as whom the priming should be conducted. If no value is provided, then the searches will be performed anonymously.

Bind Password

This specifies the password for the bind DN to use when priming the server. If no value is provided, then the searches will be performed anonymously.

Search Base

This specifies the search base that will be used when priming the server. All searches will be subtree searches below this entry.

Attribute Name

This specifies the name of the attribute to be used in priming the server. This attribute should be indexed for equality and should contain an integer value. Each entry should have a sequentially-incrementing value for this attribute.

Value Range Start

This specifies the lower bound to use when retrieving the value of the specified attribute. Entries will be retrieved in sequential order starting at this value and increasing until the value specified in the value range end parameter.

Value Range End

This specifies the upper bound to use when retrieving the values of the specified attribute.

Search Time Limit

This specifies the maximum length of time in seconds that each search will be allowed to take. Any search that takes longer than this time will be abandoned and the next search attempted.

The LDAP Authentication Jobs

One of the most common uses of LDAP directory servers is to facilitate authentications. As such, being able to test the rate at which a directory server can perform authentications is an important aspect of understanding the overall performance of the server.

SLAMD provides a number of jobs for measuring the authentication performance of a directory server. This section discusses the jobs available and the ways in which they may be used.

LDAP AuthRate

The LDAP AuthRate job provides a general simulation of the behavior of applications that perform authentications against the directory server. It will first perform a search to find a user's entry and will then bind as that user. It may also perform an additional check to determine if the user is a member of a particular static group, dynamic group, or role.

It should be noted that this job is significantly different from the **`authrate`** command-line utility provided with the Sun ONE Directory Server Resource Kit. That utility only performs bind operations (it does not first perform searches to find user entries), and it also does not provide the ability to use persistent connections. These deficiencies significantly limit the usefulness of the **`authrate`** command-line utility for measuring real-world authentication performance.

The parameters that may be configured when scheduling an LDAP AuthRate job are as follows:

Directory Server Address

This specifies the address of the directory server in which the authentications will be performed. This may be either an IP address or a resolvable name. It can also

be the address of a load balancer or directory proxy server, as long as the clients can send LDAP requests to it and receive the appropriate responses.

Directory Server Port

This specifies the port number of the directory server in which the authentications will be performed. If SSL is to be used, then this should be the server's secure port.

Directory Bind DN

This specifies the DN of the user as whom the clients will bind when performing searches to find the entries for the users that are authenticating. If no bind DN is provided, then those searches will be performed anonymously.

Directory Bind Password

This specifies the password for the bind DN. If no value is provided, then searches for user entries will be performed anonymously.

User Search Base

This specifies the DN of the entry below which all the user entries exist. All searches for user entries will be subtree searches below this entry.

Login Data File URL

This specifies the URL to a file containing information to use in the authentication process. This is an alternative to providing a single login ID value or value pattern. If a file is provided, then each line should contain information about one user, with the format of that line being the login ID value followed by a tab and the password for that user.

If a value is provided for this parameter, it should be provided as a URL. The URL may reference a file on a remote server using the HTTP protocol, like:

```
http://server.example.com/authinfo.txt
https://server.example.com/authinfo.txt
ftp://server.example.com/data/authinfo.txt
http://user:password@server.example.com/authinfo.txt
https://user:password@server.example.com/authinfo.txt
ftp://user:password@server.example.com/data/authinfo.txt
```

or a file on the local filesystem, like:

```
file:/data/authinfo.txt
```

Login ID Value

This specifies a value or value pattern that can be used to identify the user entries to use when performing the authentications. It may be a single value, but it may also contain a bracketed numeric range. For example,

```
user.[1-1000000]
```

will cause an integer value between 1 and 1000000 (inclusive) to be chosen at random and used in place of the bracketed range. It is also possible to iterate sequentially through the values by using a colon in place of the dash, like:

```
user.[1:1000000]
```

This will cause the first value chosen to be "`user.1`", the second "`user.2`", and so on. Once all 1000000 values have been exhausted, it will start back at 1. Note that the process of incrementing values sequentially can be maintained across multiple threads on the same client, but because there is no communication between the clients, each client will iterate through the same range.

Login Password

This specifies the password that will be used for authentications for users identified using the login ID value or value pattern. All users are expected to have the same password for this to work, although it is possible to configure the job to ignore bind failures because an incorrect password was provided.

Read Password from Attribute

This specifies that the password that will be used for authentications for users should be read from the indicated attribute in that user's entry. For example, a value of "`uid`" would cause the value of the user's `uid` attribute to be used as the password when performing authentications.

Login ID Attribute

This specifies the attribute containing the login ID for users. Queries to find users that are authenticating will be equality searches on this attribute with a value taken from the login ID value or from the login data file. A value must be provided for this attribute, regardless of whether the a single login ID, a login ID pattern, or a login data file is used.

Membership DN

This specifies the DN of a static group, dynamic group, or role in which to check for membership of the user that is authenticating. This is an optional parameter, and if no value is provided then no membership check will be performed. However, if a value is provided, then the user must be determined to be a member of the specified group or role for the authentication to succeed.

Warm Up Time

This specifies the length of time that the job should be running before starting to collect statistics. Using a warm up time ensures that all the clients and client threads are given an opportunity to start running and generating a consistent load against the server before statistics collection.

Note that the warm up time is considered part of the overall job duration. Therefore, if you wish to include a warm up time in a job and collect statistics for a fixed length of time, you may want to increase the overall duration accordingly.

Cool Down Time

This specifies the length of time that the job should be running before ending statistics collection. Using a cool down time ensures that all the clients and client threads are still running and generating a consistent load against the server when statistics collection ends.

Note that the cool down time will only actually be used if the job has a maximum duration or stop time. Because the cool down time is considered part of the overall job duration, you may wish to increase the duration of the job to account for the cool down time.

Operation Time Limit

This specifies the maximum length of time in seconds that any single operation (search or bind) should be allowed to take. If an operation takes longer than this length of time to perform, then the client will cancel that operation and begin on the next. If that occurs, then any authentication in process will be considered a failure.

Time Between Authentications

This parameter makes it possible to insert a delay (in milliseconds) between authentication attempts. Note that this time is measured from the beginning of one authentication attempt to to the beginning of the next, rather than between the end of one attempt and the beginning of the next. This makes it possible to target a specific number of authentications per second by adjusting the number of clients, threads per client, and delay between requests.

Use SSL

This indicates whether the authentications will be performed using SSL. This will encrypt all communication between the clients and the server, which prevents the passwords from being visible in clear text over the network. If SSL is to be used, then the port number specified should be the secure port of the directory server, and you should either indicate that the job should blindly trust any certificate presented to it or provide information about the key and/or trust stores to be used.

Blindly Trust Any Certificate

This indicates that the clients should trust any SSL certificate presented by the directory server. If this is not checked, then clients will examine the certificate presented by the certificate to determine whether it is valid and should be trusted, which will use the SSL trust store. If it is checked, then any certificate presented will be accepted so that clients do not have to have knowledge of the server's certificate or its issuer.

SSL Key Store

This specifies the location of the JSSE key store that should be used if it is necessary for the client to present a certificate to the server. Note that this job does not support the SASL EXTERNAL mechanism, so while this will allow clients to perform SSL client authentication, but it is not sufficient to allow them to use that client certificate for LDAP authentication.

SSL Key Store Password

This specifies the password needed to access information in the SSL key store. If no key store is required, then no key store password is needed.

SSL Trust Store

This specifies the location of the JSSE trust store that should be used if the client needs to verify the certificate presented by the directory server. This will be used if the client is communicating over SSL and is configured to validate the certificate presented by the server rather than blindly trust it.

SSL Trust Store Password

This specifies the password that should be used to access the contents of the JSSE trust store. This should not be needed unless the trust store is in a format that requires a password to access the information contained in it.

Number of Iterations

This specifies the maximum number of authentications that each thread should perform before it completes. A value of -1 indicates that there will not be any limit imposed. Note that if the stop time or maximum duration is reached, the job may complete before all specified iterations have been completed.

Ignore Invalid Credentials Errors

This indicates whether the client should ignore bind failures with a result of "invalid credentials" (LDAP result code 49), which is returned if an incorrect password is provided. Normally, any bind response other than "success" (LDAP result code 0) will be considered a failure of the authentication attempt. With this enabled, it is not necessary to know the passwords for all the users in order for the authentication to succeed, however it may not accurately reflect the performance of the server when a valid password is provided.

It should also be noted that an invalid credentials response does not always mean that the password provided was incorrect. For example, the Sun ONE Directory Server will also return an invalid credentials response if the user's password is expired.

Share Connections between Threads

This indicates whether the threads associated with a client should share the same set of connections to the server. If this is checked, then two connections will be established (one for searches and the other for binds) and will be shared by all threads in the client. If it is not checked, then each thread will create its own two connections to use for the searches and binds performed by that thread.

LDAP Weighted AuthRate

The LDAP Weighted AuthRate job is very similar to the LDAP AuthRate job, with the exception that it makes it possible to apply weighting to the authentication process, which means that you can target one set of users more frequently than another. This makes the job more realistic, as in a production directory it will almost certainly be the case that some users will authenticate more frequently than others. In addition, this adds another benefit in that if the directory is too large to fit entirely within the entry cache, it can still provide good performance as long as the cache is large enough to hold the set of users that authenticates most frequently.

The parameters that may be specified with the LDAP Weighted AuthRate job are very similar to the LDAP AuthRate job and therefore they will not be repeated here. However, the differences between the two are as follows:

- In the Weighted LDAP AuthRate job, it is not possible to provide files containing data to use in the authentication process. Rather, two login ID values or value patterns should be provided. Those values should be specified in the "**Login ID Value 1**" and "**Login ID Value 2**" parameters.

- The Weighted LDAP AuthRate job introduces a new configuration parameter, "**Login ID 1 Percentage**", which is used to specify the frequency with which each of the login ID value patterns is used in the authentication process.

LDAP DIGEST-MD5 AuthRate

The LDAP DIGEST-MD5 AuthRate job is virtually identical to the LDAP AuthRate job, with the exception that the authentication it performs uses the SASL DIGEST-MD5 mechanism rather than LDAP simple authentication. The DIGEST-MD5 mechanism is password based, but rather than the client providing the clear-text password to the directory server, the client sends an MD5-hashed blob of data to the server, part of which was created using the password. This makes it possible to use password-based authentication while preventing the password from traveling over the network in clear text (although it does require that the server have access to the clear-text password to verify the MD5 hash provided by the client).

Because the LDAP DIGEST-MD5 AuthRate job performs exactly the same task as the LDAP AuthRate job with only a different authentication mechanism, the configurable parameters for the two jobs are exactly the same. Therefore, they will not be repeated here.

LDAP SiteMinder Load Simulator

The LDAP SiteMinder Load Simulator job provides a mechanism for simulating the load that Netegrity® SiteMinder® places on a directory server when it is using that server to authenticate users. In particular, this job simulates the requests that SiteMinder issues to the directory server when password services are enabled.

It should be noted that this job is intended for use when tuning the directory server for use by SiteMinder clients. It should not be used as a means of trying to estimate the performance that can be achieved when using SiteMinder because it does not actually simulate the additional processing and HTTP requests that SiteMinder performs when it is performing an authentication. Therefore, while this test can help you understand the best possible performance you can expect from SiteMinder if the directory server is the bottleneck, this is often not the case and as such the number of simulated authentications per second will likely be higher than you will actually see in practice.

The parameters that may be specified when scheduling an LDAP SiteMinder Load Simulator job are similar to those that are available when scheduling an LDAP AuthRate job. For brevity, only the differences will be described here:

- The LDAP AuthRate job allows you to specify the DN of a group or role in which to check for membership of the user that is authenticating. The LDAP SiteMinder Load Simulator does not provide the ability to verify group membership as part of the authentication process.

- Part of the authentication process that SiteMinder uses is to retrieve three different attributes from the user's entry using separate searches. The contents of those attributes should not have a significant impact on performance, so any existing attributes should be acceptable. The attributes you wish to use for this should be specified in the "**First Attribute to Retrieve**", "**Second Attribute to Retrieve**", and "**Third Attribute to Retrieve**" configuration parameters.

- Another part of the SiteMinder authentication process (with password services enabled) is to modify an attribute containing metadata about the password. If you wish to perform this modification, then you should provide the name of

the attribute to modify in the "**Attributes to Modify**" parameter. If no attribute names are provided, then no modification will be performed. If you wish, you can provide multiple attributes by specifying the names of those attributes on separate lines.

LDAP Weighted SiteMinder Load Simulator

The LDAP Weighted SiteMinder Load Simulator job is very similar to the LDAP SiteMinder Load Simulator job, with the exception that it adds the ability to specify two different login ID patterns and define a weight that indicates how frequently each login ID pattern should be used when selecting the user to authenticate, and it also removes the ability to provide a login data file to use for the authentication information. The "**Login ID Value 1**" and "**Login ID Value 2**" parameters should be used to specify the patterns to use for generating the login ID values, and the "**Login ID 1 Percentage**" specifies the percentage of the time that the pattern defined by the "**Login ID Value 1**" parameter should be used to choose the login ID value.

LDAP SiteMinder Load Simulator with Replication Latency

The LDAP SiteMinder Load Simulator with Replication Latency job is similar to the LDAP SiteMinder Load Simulator job, but also adds the ability to measure the length of time required for changes made in one system to appear in another system through replication. These latency measurements are obtained by periodically performing an update in one directory (called the master) and using a persistent search in another (called the replica) to detect that change. The length of time between the completion of the change on the master and the detection of that change on the replica is recorded to the nearest millisecond.

The set of parameters available for this job are slightly different from those available for the LDAP SiteMinder Load Simulator job. In particular:

- This job does not support the use of SSL.
- The "**Directory Server Address**" and "**Directory Server Port**" parameters have been renamed to "**Master Directory Host**" and "**Master Directory Port**", respectively. These specify the information used to connect to the master directory, in which the changes will be made.

- New "**Replica Directory Host**" and "**Replica Directory Port**" parameters have been added that can be used to provide the information needed to connect to the replica directory server that will be used to detect the changes made in the master.

- A new "**Latency Check Entry DN**" parameter has been added that can be used to specify the DN of the entry that should be periodically modified on the master, and against which the persistent search will be registered on the replica server. Note that this entry should be one that is not modified as part of the normal SiteMinder authentication process, nor should it be modified by anything other than the latency checking process itself. Any updates to this entry not performed by the latency checking process will interfere with the accuracy of the latency check measurements.

- A new "**Time Between Latency Checks (ms)**" parameter has been added that specifies the minimum length of time in milliseconds to allow between updates to the latency check entry. If it takes less time than specified by this parameter for a change to be propagated from the master to the replica, then the latency checking process will sleep for the remainder of that time before performing the next modification against the latency check entry. If it takes more than this time for the change to be replicated, then there will be no delay before the next update attempt.

LDAP Weighted SiteMinder Load Simulator with Replication Latency

The LDAP Weighted SiteMinder Load Simulator with Replication Latency job adds the ability to measure replication latency to the LDAP Weighted SiteMinder Load Simulator job. The changes associated with this job are the same as the set of changes between the LDAP SiteMinder Load Simulator and LDAP SiteMinder Load Simulator with Replication Latency jobs (i.e., no SSL support, parameters for both master and replica host and port, and latency check entry DN and latency check delay).

LDAP Solaris Authentication Load Generator

The LDAP Solaris Authentication Load Generator job provides a means of simulating the load that Solaris Native LDAP II (as performed by Solaris 9)

places on the directory server when configured to use PAM_LDAP. More specifically, it simulates the load that Solaris places on the directory server whenever a client authenticates to the system via telnet.

As with the SiteMinder job mentioned above, this job should not be considered a simulation of the Solaris authentication process itself, but rather a simulation of the load that it places on the directory server. Solaris performs additional processing during an authentication other than simply issuing requests to the directory server, and therefore the actual performance that can be achieved may be lower than what is reported by this job.

This job contains a number of configurable parameters that can customize the way that the authentication process will be handled. Many of these parameters correspond to options that may be specified when running the **`idsconfig`** or **`ldapclient`** utilities to prepare a system to be used as an LDAP client. Those options include:

Directory Server Address

This specifies the address of the directory server that will be used when performing authentications. This corresponds to the "**iPlanet Directory Server's (iDS) hostname**" option when running **`idsconfig`**. It may be either an IP address or a fully-qualified resolvable name. The address specified can also be the address of a load balancer or directory proxy server, as long as the clients can send LDAP requests to it and receive the appropriate LDAP responses.

In order for the authentications to succeed, this directory server should be populated with Solaris naming data. At least the passwd, shadow, and hosts repositories should have been added to the server. This does not need to be real user data (in fact, it is easier to test with generated data), and the MakeLDIF **`solaris.template`** template should be sufficient for doing this.

Directory Server Port

This specifies the port number that clients should use to communicate with the directory server. This corresponds to the "**port number for iDS**" option when running **`idsconfig`**. If clients are to communicate with the directory server over SSL, then this should be the secure port of the directory server.

Directory Base DN

This specifies the DN of the entry below which all the Solaris naming information exists in the directory. This corresponds to the "**LDAP Base DN**" option when running **`idsconfig`**.

Credential Level

This specifies the credential level that will be used when performing operations in the directory server, and it corresponds to the "**Credential level**" option when running **`idsconfig`**. Note that this job only supports the "anonymous" and "proxy" credential levels. The "proxy anonymous" level is not supported. If the "proxy" credential level is used, then you will also need to provide a proxy user DN and password.

Proxy User DN

This specifies the DN of the user that the Solaris clients will use when binding to the directory server in order to submit requests required for authentication. This is only required if a credential level of "proxy" is specified. The DN of this user is typically something like "**`cn=proxyagent,ou=profile,`***{directory-base-dn}*", where "*{directory-base-dn}*" is the value provided for the Directory Base DN configuration parameter.

Proxy User Password

This specifies the password for the proxy user. It will only be required if the "proxy" credential level has been selected.

Authentication Method

This specifies the method that will be used to verify the credentials provided by the end user. It may have one of the following values:

- Simple Authentication -- This will use a standard LDAP simple bind to authenticate to the directory server.
- DIGEST-MD5 Authentication -- This will perform a bind using the SASL DIGEST-MD5 mechanism. This will prevent the password from traveling over the network in clear text, but requires that the server have access to the clear-text password for the user.
- Simple Authentication over SSL -- This will use a standard LDAP simple bind to authenticate to the directory server, but all communication with the server will be encrypted using SSL.
- DIGEST-MD5 Authentication over SSL -- This will perform a bind using the SASL DIGEST-MD5 mechanism, and all communication with the server will be encrypted using SSL.

Note that the same mechanism for communicating with the server will be used for performing binds and also for all the other lookups associated with the authentication process. If either "Simple Authentication over SSL" or "DIGEST-MD5 Authentication over SSL" are chosen, then the port number provided should be the secure port of the directory server.

User ID

This specifies the user ID to use when performing the authentication. This can be a single user ID, but it can also contain a numeric range enclosed in brackets. For example, the user ID

```
user.[1-1000000]
```

will cause the client to choose a number at random between 1 and 1000000 (inclusive), and the client will use that number in place of the bracketed range. Similarly, a user ID of

```
user.[1:1000000]
```

will cause the client to iterate through these values starting at 1 for the first request and increasing sequentially for each subsequent request. Once all 1000000 values have been exhausted, it will start back at 1. Note that if this sequential method is used, it can only be preserved across a single client, although it will be preserved across multiple threads within that client.

User Password

This specifies the password to use when binding to the server as the end user. For the purposes of this test, it is expected that all users will have the same password.

Simulated Client Address Range

This specifies the address or address range from which clients will appear to be authenticating to the directory. When a client attempts to authenticate to a Solaris system configured as an LDAP client, a **`gethostbyaddr`** system call will be used in an attempt to resolve the address of that client, which will translate into a request to the directory server.

The value of this parameter may be a single IPv4 address, or it may indicate a range of addresses using the CIDR address scheme as defined in RFC 1519. In a CIDR address range, a base address is provided followed by a forward slash and the number of bits that are considered significant. For example, a CIDR address

of "192.168.1.0/24" indicates that any address in that range will start with the same 24 bits as "192.168.1.0", which is effectively "192.168.1.[0-255]".

Blindly Trust Any Certificate

This indicates whether the client should blindly trust any certificate presented by the directory server if it wishes to communicate over SSL. If this is checked, then the directory server's certificate will be considered trusted regardless of whether the client has any knowledge of that certificate or its issuer. If it is unchecked, then the client will attempt to validate the certificate provided by the directory server.

SSL Key Store Location

This specifies the location of the JSSE key store that will be used if the client must present a certificate to the directory server when authenticating. Note that this job does not support the SASL external mechanism, and therefore even if a client certificate is required, that certificate will not be used to perform LDAP authentication.

SSL Key Store Password

This specifies the password required to access the information in the JSSE key store.

SSL Trust Store Location

This specifies the location of the JSSE trust store that will be used to determine whether the client should trust the certificate presented by the directory server. This is not needed if the communication is not encrypted using SSL, or if the client is configured to blindly trust any certificate that the directory server may provide.

SSL Trust Store Password

This specifies the password that should be used to access the information in the JSSE trust store. It should only be used if the JSSE trust store is needed and if it is stored in a format that requires a password to access the information contained in it.

The LDAP Modify Jobs

While the majority of operations in a typical directory are read operations (e.g., search, bind, and compare), the speed with which the server can process write operations is also important to understand because it can have an impact on the overall performance of the server. There are four main types of write operations that can be performed:

- Modify operations. SLAMD jobs for measuring modify performance will be discussed in this section.
- Add and delete operations. SLAMD jobs for measuring add and delete performance will be discussed in the next section.
- Modify DN operations. These are much less common in a typical directory environment and SLAMD does not provide any jobs dedicated to performing modify DN operations. However, the LDAP Load Generator jobs do provide the capability to perform a variety of LDAP operations, including modify DN.

LDAP ModRate

The most basic job that SLAMD offers for measuring the performance of modify operations is the LDAP ModRate job. This job offers the ability to replace the value of a specified attribute with a randomly-generated value of a designated length. It is similar to the **`modrate`** command-line utility provided in the Directory Server Resource Kit (DSRK), but offers support for additional features like SSL communication and rate limiting.

The parameters that may be specified when scheduling an LDAP ModRate job include:

Directory Server Host

This specifies the address (either IP address or resolvable name) of the directory server in which the modifications are to be performed. In order for the modifications to succeed, then either the server should be writeable or the client should be configured to follow referrals.

Directory Server Port

This specifies the port number that clients should use for communicating with the directory server. If SSL is to be used to perform the modifications, then this should be the secure port of the server.

Bind DN

This specifies the DN that should be used when binding to to the directory server. This user should have permission to perform modify operations in the server unless proxied authorization is to be used, in which case this user must have permission to use the proxied authorization control.

Bind Password

This specifies the password that should be used when binding to the directory server.

Proxy As DN

This indicates whether the proxied authorization control should be used when performing the modifications, and if so the DN of the user as whom the modifications should be performed. If no value is provided, then the proxied authorization control will not be used and the modifications will be performed using the identity of the bind DN. If a value is provided, then it should be the DN of the user as whom the modify operations are to be performed, and the bind DN should be the DN of a user that has permission to use the proxied authorization control.

Entry DN

This specifies the DN or DN pattern to use when determining the entries to modify. It can be a single DN, but it can also include a bracketed numeric range to indicate multiple entries. For example, the value:

```
uid=user.[1-1000000],ou=People,dc=example,dc=com
```

can be used to indicate that a number should be chosen at random between 1 and 1000000 (inclusive) for each request and used in place of the bracketed range to construct the DN of the entry to modify. Similarly, replacing the dash with a colon, like:

```
uid=user.[1:1000000],ou=People,dc=example,dc=com
```

will cause the numbers to be chosen in sequential order rather than at random. Note that if the sequential ordering is used, it will only work properly on a single client because no attempt is made to enforce any kind of communication between the clients. However, the ordering will be handled correctly for multiple threads on a single client.

DN File URL

This specifies the URL to a file containing the DNs of the entries to modify. This is an alternative to providing a single entry DN or DN pattern, and if a DN file is provided then you should not use an entry DN or DN pattern.

If a file is to be used, then it should contain one entry DN per line. The location of that file should be specified using a URL. The file can be located on a remote server that the clients will access using the HTTP protocol by specifying a URL like:

```
http://server.example.com/dns.txt
https://server.example.com/dns.txt
ftp://server.example.com/data/dns.txt
http://user:password@server.example.com/dns.txt
https://user:password@server.example.com/dns.txt
ftp://user:password@server.example.com/data/dns.txt
```

or it can specify a file on the local filesystem with a URL like:

```
file:/data/dns.txt
```

Attribute to Modify

This specifies the name of the attribute that should be targeted by the modify requests. The modify operation will replace the value of this attribute with a string of randomly-chosen alphabetic characters.

Value Length

This specifies the number of characters to include in the new value when performing the modify operations. This can be significant if the attribute to

modify is indexed, and it can be especially significant if that attribute is configured with a substring index.

Warm Up Time

This specifies the length of time that the job should be allowed to run before beginning to collect statistics. Using a small warm up time allows time for all the clients and client threads to begin running and generating a consistent load against the directory server before beginning to capture statistics.

Note that the warm up time is included in the overall job duration. Therefore, if you use a warm up time, then you may wish to increase the job duration accordingly so that the desired length of time is spent actually capturing statistics.

Cool Down Time

This specifies the length of time that the job should continue to run after ending statistics collection. This helps ensure that all the clients and client threads are still active when ending statistics collection so that the load against the directory server is still consistent. Note that a cool down time can only be used if the job is scheduled to end because of a maximum duration or stop time. It will not use a cool down time if it is stopped because it has reached the maximum number of iterations or because it has been cancelled by an administrator.

You should also note that the cool down time is included in the overall job duration. Therefore, if you wish to have a cool down time, then you should increase the duration of the job to account for this cool down time.

Modify Time Limit

This specifies the maximum length of time in second that a modify operation should be allowed to take. If the operation takes longer than this length of time to complete, then the client will abandon the request and start on the next modification.

Time Between Requests

This specifies the minimum length of time in milliseconds that should be allowed between requests. This time is measured from the beginning of one modify request to the beginning of the next, rather than between the end of one request and the beginning of the next. This makes it possible to target a particular number of modify operations per second. Note that if any single modify

operation takes longer than this length of time to complete, there will not be any delay before the next request is issued.

Use SSL

This indicates whether communication between the clients and the directory server should be encrypted using SSL. If SSL is to be used, then the port number provided for the server should be the secure port.

Blindly Trust Any Certificate

This indicates whether the clients should blindly trust any certificate presented by the directory server if they are communicating over SSL. If this is not enabled, then the client will use the information in the SSL trust store to determine whether the certificate should be trusted.

SSL Key Store

This specifies the location of the JSSE key store that will be used if it is necessary for the client to present a certificate to the directory server. Note that even if the directory server is configured to require SSL client authentication, that certificate will not be used for LDAP authentication because this job does not provide support for the SASL EXTERNAL mechanism.

SSL Key Store Password

This specifies the password that should be used to access the information in the JSSE key store.

SSL Trust Store

This specifies the location of the JSSE trust store that should be used to determine whether to trust the SSL certificate presented by the directory server.

SSL Trust Store Password

This specifies the password that should be used to access the information in the JSSE trust store. Note that this may not be required if the JSSE trust store does not require a password in order to read the information it contains.

Number of Iterations

This specifies the number of modify operations that each client thread should perform before exiting. The total number of modifications performed by the job will be the product of the number of clients, the number of threads per client, and the number of iterations per thread.

Always Disconnect

This indicates whether the client should use a separate connection to the directory server for each modify operation. By default, each client thread will establish one connection at the beginning of the job and will use that connection for all modifications performed. If this parameter is checked, however, the client will establish a connection, bind, perform the modify operation, and unbind for each operation.

Follow Referrals

This indicates whether the client will follow any referrals they may encounter while performing modify operations. By default no referrals will be followed.

LDAP Weighted ModRate

The LDAP Weighted ModRate job is very similar to the LDAP ModRate job, but it offers the capability to target two sets of entries when performing the modifications, and to use different frequencies when accessing each set. For example, 80% of the operations could be targeted at a relatively small portion of the data set, and the remaining 20% could be targeted at the remainder of the data.

Because the LDAP Weighted ModRate job is very similar to the LDAP ModRate job, only the differences in configurable parameters will be discussed here. Those differences include:

- The LDAP ModRate job offers the capability to provide a file containing the DNs of the entries in which the modify operations are to be performed. This is not available with the LDAP Weighted ModRate job.
- You may provide two different entry DN or DN patterns when performing the modifications. These should be provided in the "**Entry DN 1**" and "**Entry DN 2**" parameters.

- An additional configuration parameter, "**DN 1 Percentage**" is used to specify how frequently the value of the "**Entry DN 1**" parameter is used to choose the DN of the entry to modify. The percentage of the time the value of the "**Entry DN 2**" parameter is used to choose the DN will be 100 minus the value of this parameter.

LDAP ModRate with Replica Latency

This job is very similar to the LDAP ModRate job, but it has a very useful feature for operating in replicated environments. It provides the ability to measure the length of time required for a change to replicate from one server to another. It does this by periodically making changes to a specific entry on one system and using a persistent search to determine when that change has been replicated to the other server.

Because the LDAP ModRate with Replica Latency job is very similar to the LDAP ModRate job, only the differences in configuration parameters will be discussed here. Those differences include:

- The LDAP ModRate job has the ability to follow referrals if they are encountered during the modification process. The LDAP ModRate with Replica Latency job will not follow any referrals encountered while performing the modifications.

- Rather than providing a single address for the directory server, there are two addresses required. The "**Master Directory Host**" is the address to use for the master directory server, on which all changes will be made. The "**Replica Directory Host**" is the address of the server that will receive replication changes from the master directory server. Note that these must be the addresses of the directory server systems themselves and should not be addresses that are load balanced or may resolve to multiple systems. The master directory server must be writeable and should not send referrals.

- The directory server ports have also been specified separately. Rather than a single port number, you should provide separate values in the "**Master Directory Port**" and "**Replica Directory Port**" configuration parameters.

- A new configuration parameter, "**Latency Check Entry DN**" is available. This specifies the DN of the entry that will be monitored to measure how quickly changes are replicated from the master server to the replica. This entry should exist in the same database as the rest of the entries that will be modified, but it should not be included in the set of entries to modify listed in

either the DN file or using the entry DN or DN pattern. Further, this entry should not be modified by any external applications while the job is running.

- A new configuration parameter, "**Time Between Latency Checks**" is available. This specifies the length of time in milliseconds between modifications to the entry specified by the "**Latency Check Entry DN**". Note that this length of time is configured as the length from the beginning of one request to the beginning of the next, rather than the end of one to the beginning of the next. If it takes longer than this length of time to perform the modification and for that change to propagate to the replica, then no delay will be inserted before the next request.

LDAP Weighted ModRate with Replica Latency

The LDAP Weighted ModRate with Replica Latency job, as its name implies, is a combination of the LDAP Weighted ModRate and the LDAP ModRate with Replica Latency jobs. Essentially, it is identical to the LDAP ModRate with Replica Latency job, but has the ability to target two sets of entries with different frequencies. Because this job is similar to the two jobs earlier discussed, the parameters available for scheduling this type of job will not be discussed here.

The LDAP Add and Delete Jobs

Although it is relatively simple to measure the modify performance of an LDAP directory server, measuring the performance of add and delete operations is somewhat more complex because they can be considered destructive operations. That is, the directory is in a different state after the operation has completed than before, and the same test cannot be run again without returning to the original state, either by restoring a backup, removing entries that have been added, or adding entries that have been deleted.

SLAMD does provide jobs that perform add and delete operations in this destructive manner, namely the LDAP AddRate job and the LDAP DelRate job. However, they can work in conjunction with each other (e.g., the LDAP AddRate job can be used measure the performance of adding a number of entries to the server, and then the LDAP DelRate job can be used to measure the performance of deleting those entries) so that the net result is a directory that is essentially in the same state after the tests as before. However, SLAMD also provides a job, the LDAP Add and Delete Rate job, that combines both add and delete operations into a single job. This is much more useful because it makes it possible to use optimizing jobs to measure the performance of add and delete operations, rather than being required to interleave individual LDAP AddRate and LDAP DelRate jobs.

LDAP AddRate

The LDAP AddRate job can be used to measure the performance of the directory server when performing LDAP add operations. It does this by adding entries to the server whose RDN value is a sequentially increasing integer value and whose attributes consist of randomly generated alphabetic strings.

It is very important to note that because the LDAP AddRate job uses a sequentially-increasing number as the RDN value for the entries that are added, it is not possible to use multiple clients when scheduling an LDAP AddRate job. If multiple clients were used, only one of them would actually add each entry and

the others would receive error messages (in particular, LDAP result code 68, entry already exists) when trying to add the same entry. The LDAP AddRate job should not allow itself to be scheduled with multiple clients, and any attempt to do so will result in an error message stating that only a single client may be used. However, it is possible to use multiple threads on a single client to perform multiple add operations concurrently.

By default, the following entries will be generated with the following attributes:

- **objectClass**
- **givenName**
- **sn**
- **cn**
- **uid**
- **mail**
- **userPassword**

The values provided for the **objectClass** attribute will be:

- **top**
- **person**
- **organizationalPerson**
- **inetOrgPerson**
- **extensibleObject**

It is not possible to exclude any of these attributes, or to change the set of **objectClass** values that will be used. However, it is possible to add additional attributes to the entries that will be generated, and because **extensibleObject** is used in the **objectClass** definitions, it is possible to include attributes that would not otherwise be allowed by the schema definitions in the server.

For this job, it is recommended that in most cases the job be allowed to run until it has added all entries for which it has been configured. If the job stops due to a maximum duration or stop time, then it will not have added all entries for which it has been configured and it may not be easy to determine exactly how many entries have been added so that they can be subsequently removed. Therefore, it is recommended that you not specify a stop time or maximum duration unless the directory is to be returned to its original state by restoring a backup rather than by deleting entries.

The following attributes may be specified with the LDAP AddRate job:

Directory Server Host

This specifies the address of the directory server in which the entries are to be added. It may be either an IP address or a resolvable name. It may also be the address of a load balancer or LDAP proxy server, although if that is the case then all servers to which the adds will be sent must be able to process write operations

(this job does not offer the capability to follow referrals). It should also be noted that attempting to balance the load across multiple servers may alter the performance characteristics of the server because it may introduce increased contention as a result of replication.

Directory Server Port

This specifies the port number of the directory server in which the entries are to be added. If SSL is to be used, then this should be the secure port of the directory server.

Bind DN

This specifies the DN of the user as whom the clients will bind in order to perform the add operations. If proxied authorization is to be used, then this user must have permission to use the proxied authorization control. Otherwise, this user must have permission to perform the requested add operations.

Bind Password

This specifies the password that should be used when binding to the directory server.

Proxy As DN

This indicates whether proxied authorization should be used when adding entries to the server, and if so specifies the DN of the user under whose authority the adds should be performed. If no value is provided for this parameter, then the proxied authorization control will not be used and the specified bind DN will be used to perform the add operations. If a value is provided for this parameter, then it should be the DN of the user as whom the adds will be performed. If this is used, then the user specified in this value must have permission to perform the requested add operations, and the bind DN must have permission to use the proxied authorization control.

Base DN

This specifies the DN of the entry below which the new entries will be created in the server. This entry must exist in the directory for the add operations to succeed.

RDN Attribute

This specifies the name of the attribute that will be used as the RDN attribute for the entries that are created. Note that this job does not support the use of multivalued RDNs when creating entries.

Initial RDN Value Number

This specifies the integer that will be used as the RDN value of the first entry created by this job. Subsequent entries added will have an RDN value that is one higher than the previous value.

Final RDN Value Number

This specifies the integer that will be used as the RDN value of the last entry created by this job. Entries created will start with an RDN value equal to the value of the **Initial RDN Value Number** parameter, and will increase sequentially until reaching the value of this parameter, at which point the job will be complete.

Generated Value Length

This specifies the number of characters that will be used for each attribute value generated when creating the entries to add. Note that this can have a significant impact on the overall performance of the job, particularly if any of the attributes in the entries to be added include substring indexes.

Additional Attributes

This specifies a set of additional attributes that may be included in the entries that are generated. The new entries will always contain the attributes listed earlier, but it is possible to include additional attributes by specifying them in this parameter. If multiple additional attributes are to be included, then a separate attribute name should be provided per line. Note that it is also possible to provide explicit values for any additional attribute created by following the name of that attribute with a colon, a space, and the desired value.

Warm Up Time

This specifies the length of time in seconds that the job will be allowed to run before the client begins collecting statistics. This makes it possible for all client

threads to be started and generating a consistent load on the server before starting to collect statistics.

Cool Down Time

This specifies the length of time in seconds that the job should be allowed to run after statistics collection ends. This makes it possible for all client threads to be active and generating a consistent load on the server when statistics collection ends so that the results will not be skewed by an inconsistent load on the server during the time in which the job is ending.

Note that the cool down time will only be used if the job is configured to stop because it has run for a maximum allowed duration or the stop time has been reached. If the job ends because it has added all entries for which it has been configured, then no cool down time will be used.

Add Time Limit

This specifies the maximum length of time in seconds that any add operation should be allowed to take. If any single add operation takes longer than this length of time, then it will be abandoned and the client will proceed with the next entry.

Time Between Requests

This specifies the minimum length of time in milliseconds that should be used between add requests. Note that this time will be measured between the beginning of one request and the beginning of the next, rather than between the end of one request and the beginning of the next. Measuring the time in this manner makes it possible to target a particular number of add operations per second by adjusting the delay between requests and the number of client threads that will be used. It should also be noted that if any add operation takes longer than this length of time to complete, there will be no delay before the next request.

Use SSL

This indicates whether the client should use SSL when communicating with the directory server. If SSL is to be used, then the port provided should be the secure port of the directory server.

Blindly Trust Any Certificate

This indicates whether the client should blindly trust any certificate that the server may present when communicating over SSL. If this is not checked, then the client will use the information contained in the SSL trust store to determine if the server certificate should be trusted.

SSL Key Store

This specifies the location of the JSSE key store that will be used if the client needs to present a certificate to the directory server. Note that this job does not support the SASL external mechanism, however, so even if a client certificate is provided to the server it may not be used for LDAP authentication.

SSL Key Store Password

This specifies the password that should be used to access the information in the JSSE key store.

SSL Trust Store

This specifies the location of the JSSE trust store that will be used to determine whether the client should trust the SSL certificate presented by the directory server. This will be used if the client is to communicate over SSL but has not been configured to blindly trust any SSL certificate.

SSL Trust Store Password

This specifies the password that should be used to access the information in the JSSE trust store. It is unlikely that this will be required unless the trust store is in a format that requires a password for read-only access to the information that it contains.

Always Disconnect

This indicates whether the client should use a separate connection to the directory server for each add operation, or whether the client should use persistent connections. By default, each client thread will establish a connection to the server at the beginning of the job and will use that connection for all add operations performed by the thread. If this option is checked, however, then the client will establish a connection, bind, perform an add, and disconnect for each entry that needs to be added.

LDAP DelRate

The LDAP DelRate job can be used to measure the performance of the directory server when performing delete operations. It is largely similar to the LDAP AddRate job, with the exception that it performs delete operations instead of adds. In fact, it is well-suited to remove the entries added by the LDAP AddRate job, although it can actually be used for other purposes as well.

Because the set of configurable options for this job are very similar to those available for the LDAP AddRate job, they will not be repeated here. Rather, only the differences in configurable parameters will be discussed. Those differences are discussed below. Note that like the LDAP AddRate job, only one client may be used with the LDAP DelRate job.

The first major difference is that rather than providing an RDN attribute, initial value number, and final value number, the LDAP DelRate job allows you to specify a DN pattern to use. This is a rather flexible approach and can be used to delete entries created by the LDAP AddRate job, but may also be used for other kinds of entries as well. The "**Entry DN Pattern**" parameter allows you to specify an entry DN that includes a bracketed numeric range. For example, providing a DN pattern of:

```
uid=user.[1:5000],ou=People,dc=example,dc=com
```

will cause the job to delete a total of 5000 entries (with **`uid`** values of **`user.1`**, **`user.2`**, **`user.3`**, ..., **`user.5000`**) in sequential order. Note that unlike other jobs that accept a bracketed numeric range, it is not possible to use a dash instead of a colon to choose the entries at random rather than sequentially because as the number of delete operations performed by the job increases, the likelihood of trying to delete entries that no longer exist also increases.

Another difference is that you can delete entries with arbitrary DNs by providing those DNs in a file (one entry DN per line). The location of this file should be specified using the "**DN File URL**" parameter, and it should actually be a URL that provides the location of the file. This URL may reference a file on a remote server that is accessible over the HTTP protocol, like:

```
http://server.example.com/dns.txt
https://server.example.com/dns.txt
ftp://server.example.com/data/dns.txt
http://user:password@server.example.com/dns.txt
https://user:password@server.example.com/dns.txt
ftp://user:password@server.example.com/data/dns.txt
```

or it may reference a file on the local filesystem, like:

```
file:/data/dns.txt
```

The entries will be deleted in the order that they are provided in the data file. Once all entries in the file have been removed, the job will complete.

LDAP Add and Delete Rate

The LDAP Add and Delete Rate job is a very useful job because it combines the process of adding and deleting entries, but it measures the performance of each separately. In addition, it the job is allowed to run to completion, then the directory should be in essentially the same state as it was before the job ran because all entries that were added would have also been deleted.

The configuration parameters for the LDAP Add and Delete Rate job are almost exactly the same as for the LDAP AddRate job class, and therefore they will not be discussed here. However, the LDAP Add and Delete Rate job does offer a couple of additional parameters.

The "**Time Between Adds and Deletes**" parameter specifies the delay in seconds that the client should wait after completing the adds before starting on the delete operations. This will make it possible for the directory server to return to a largely idle state after flushing all changes to the database so that processing associated with the add operations does not impact the performance of the delete operations.

The "**Clean Up Tombstones**" parameter is particularly useful when testing on a Sun ONE Directory Server with replication enabled because it provides the ability to remove any tombstones that remain in the database after performing a number of delete operations. Note that this should never be used in a production server because it can interfere with the ability of the server to resolve replication conflicts. However, it can be important for a benchmark in which many deletes are performed because otherwise the size of the database may continue to grow, which will impact the ability of the server to cache all database contents.

The "**Delay Before Cleaning Tombstones**" parameter specifies the length of time in seconds after all deletes have completed that the job should wait before beginning the process of cleaning up tombstones.

AddRate, DelRate, and Add and Delete Rate with Replica Latency

The LDAP AddRate with Replica Latency, LDAP DelRate with Replica Latency, and LDAP Add and Delete Rate with Replica Latency are exactly like their counterparts discussed earlier in this section, but they also provide the ability to measure the latency associated with replication while add and/or delete operations are in progress. It does this in the same way as the LDAP ModRate with Replica Latency job, which is by establishing a persistent search against a specific entry on the replica and tracking how long it takes for periodic changes to that entry on the master to be applied on the consumer.

Most of the parameters for these jobs are exactly the same as for the versions of these jobs that do not measure replication latency. However, each of them does have an additional set of parameters to use when measuring replication performance. Those additional configuration parameters are:

Replica Directory Host

This specifies the address of the replica directory server that will receive changes applied to the master directory server. It may be an IP address or a resolvable name, but it must resolve to a single system and may not be load balanced.

Replica Directory Port

This specifies the port number of the replica directory server that will receive changes applied to the master directory server. Note that if communication with the master directory server is to be over SSL, then SSL will also be used to communicate with the replica. If SSL communication is to be used, then this should be the secure port of the replica directory.

Latency Check Entry DN

This specifies the DN of the entry that will be periodically modified on the master server, and that will also be watched for updates using a persistent search on the replica. Note that even though the majority of the operations against the master server will be add or delete operations, the latency check operations will still be modifies because they are easier to track in a reliable manner and it should not have an impact on performance because changes are replicated in the order in which they were performed on the master server, regardless of the type of operation. Note that this entry should be in the same database as the entries that are being added and/or deleted, and it should not be modified by anything other than the latency checking process while the job is active.

Time Between Latency Checks

This specifies the length of time in milliseconds that should be allowed between latency checks. This makes it possible to target a specific latency check rate, although if any change requires longer than this time to be replicated, then there will be no delay before the next latency check is started.

Attribute to Modify

This specifies the address of the attribute to modify when performing the latency checking operations.

Template-Based AddRate

The Template-Based LDAP AddRate job is a powerful alternative to the LDAP AddRate job that provides additional flexibility and control when defining the entries to be added. Rather than using a default inetOrgPerson entry like the LDAP AddRate job, this job allows the user to provide a template that describes in detail the structure to use for the entry.

The job is provided with a default template that should be suitable in many cases. However, if it is not suitable, then it may be easily modified before submitting the job. The customization for each entry created may be performed by embedding tags in the template itself. Those tags include the following:

- **<presence:***{percent}***>** -- This tag may be used to control the percentage of entries in which the associated attribute value exists. Note that "*{percent}*" should be replaced with an integer value between 0 and 100 when using this tag.

- **<ifpresent:***{attr}***>** -- This tag may be used to make one attribute value dependent upon another. If the attribute "*{attr}*" exists in the entry being created, then the attribute value containing this **ifpresent** tag will also be included in the entry.

- **<ifpresent:***{attr}***:***{value}***>** -- This tag may be used to make one attribute value dependent upon another. If the attribute "*{attr}*" exists with a value of "*{value}*" in the entry being created, then the attribute value containing this **ifpresent** tag will also be included in the entry.

- **<ifabsent:***{attr}***>** -- This tag may be used to make one attribute value dependent upon another. If the attribute "*{attr}*" does not exist in the entry being created, then the attribute value containing this **ifpresent** tag will be included in the entry.

- **<ifabsent:***{attr}***:***{value}***>** -- This tag may be used to make one attribute value dependent upon another. If the attribute "*{attr}*" does not have a value of "*{value}*" in the entry being created, then the attribute value containing this **ifpresent** tag will be included in the entry.

- **<entrynumber>** -- This tag will be replaced with the entry number assigned to the current entry. The first entry created will have an entry number specified by the "Initial Entry Value Number" parameter, and each entry created after that will use a sequentially greater value until the number specified by the "Final Entry Value Number" parameter is reached.

- **<random:chars:***{charset}***:***{length}***>** -- This tag will be replaced with a string of *{length}* characters chosen at random from the character set *{charset}*.

- **<random:alpha:***{length}***>** -- This tag will be replaced with a string of *{length}* characters chosen at random from the set of lowercase alphabetic characters.

- **<random:numeric:***{length}***>** -- This tag will be replaced with a string of *{length}* characters chosen at random from the set of numeric digits.

- **<random:numeric:***{minvalue}***:***{maxvalue}***>** -- This tag will be replaced with a randomly chosen integer value between the integers *{minvalue}* and *{maxvalue}*.

- **<random:alphanumeric:***{length}***>** -- This tag will be replaced with a string of *{length}* characters chosen at random from the set of lowercase alphabetic characters and numeric digits.

- **<random:hex:***{length}***>** -- This tag will be replaced with a string of *{length}* characters chosen at random from the set of hexadecimal characters (all numeric digits and all lowercase characters in the range "a" through "f").

- **<random:base64:***{length}***>** -- This tag will be replaced with a string of *{length}* characters chosen at random from the set of base-64 characters (all uppercase and lowercase alphabetic characters, all numeric digits, and the "+" and "/" symbols). Note that as per the base-64 specification, the value created may be padded with equal signs so that the length is a multiple of four characters.

- **<random:telephone>** -- This tag will be replaced with a randomly-chosen telephone number in the format "xxx-xxx-xxxx" (in which each "x" will be replaced by a randomly-chosen numeric digit).

- **<random:month>** -- This tag will be replaced with the full name of a randomly chosen month.

- **<random:month:***{length}***>** -- This tag will be replaced with at most *{length}* characters from the full name of a randomly-chosen month.

- **<guid>** -- This tag will be replaced with a value in the format of a globally-unique identifier (i.e., "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx", in which each "x" will be replaced by a randomly chosen hexadecimal digit).

- **<sequential>** -- This tag will be replaced with a sequentially-incrementing numeric value. It will have a value of 0 for the first entry created.

- **<sequential:***{startvalue}***>** -- This tag will be replaced with a sequentially-incrementing numeric value. It will have a value of *{startvalue}* for the first entry created.

- **<base64:***{value}***>** -- This tag will be replaced with the base-64 representation of the string *{value}* using the UTF-8 encoding.

- **<base64:***{encoding}***:***{value}***>** -- This tag will be replaced with the base-64 representation of the string *{value}* using the *{encoding}* encoding.

In addition to the tags listed above, it is also possible to include the value of a previously-defined attribute by placing the name of that attribute in curly braces. For example, in the template line:

```
mail: {uid}@example.com
```

the **mail** attribute will be constructed from the concatenation of the value of the **uid** attribute and the static string "**@example.com**". It is also possible to specify a maximum number of characters to take from the value of the specified attribute by placing a colon and the indicated number of characters after the name of the attribute inside the curly braces. For example, the template line:

```
initials: {givenName:1}{sn:1}
```

will construct the value of the **initials** attribute from the first character of the **givenName** attribute and the first character of the **sn** attribute.

The default template used for this job is:

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
employeeNumber: <entrynumber>
```

```
uid: user.{employeeNumber}
givenName: User
sn: {employeeNumber}
cn: User {sn}
initials: {givenName:1}{sn:1}
mail: {uid}@example.com
userPassword: password
telephoneNumber: <random:telephone>
mobile: <random:telephone>
pager: <random:telephone>
homePhone: <random:telephone>
```

The configuration parameters that may be specified when using a template-based AddRate job are very similar to the parameters that may be specified when scheduling an LDAP AddRate job. The only differences are:

- The optional parameter that allows the user to specify a set of additional attributes to include in the entry is no longer available. It has been replaced by a required attribute that defines the template to use to create the entries.

- The "Initial RDN Value Number" and "Final RDN Value Number" parameters have been replaced by "Initial Entry Value Number" and "Final Entry Value Number", respectively. The new parameters serve two purposes: to help control the number of entries that are created, and to supply values for the "**<entrynumber>**" tag that may be used in the template for creating entries.

Other LDAP Jobs

SLAMD was originally designed for the purpose of measuring the performance of LDAP operations, although great care was taken during the development process to ensure that it remained a general purpose tool capable of interacting with any kind of network application. However, because of the focus on LDAP operations, many of the jobs provided with SLAMD are targeted at interacting with directory servers. Previous sections have discussed some of those jobs, but there are still other jobs provided that do not fit into any of the previous categories. Those jobs are discussed in this section.

ldif2db Import Rate

The ldif2db Import Rate job may be used to execute the ldif2db utility provided with the Sun ONE Directory Server to import LDIF data into a specified database. While the import is in progress, statistics will be collected about the average and recent import rate and the total number of entries imported.

This job is technically not an LDAP job because it does not communicate using the LDAP protocol. However, it is intended for use in conjunction with the Sun ONE Directory Server, and therefore it will be categorized as an LDAP job for the purposes of this document, and also in SLAMD.

The parameters available for this job are as follows:

ldif2db Command

This specifies the absolute path to the **`ldif2db`** utility provided with the Directory Server. This should be the path to the **`ldif2db`** utility for the instance in which the import will be performed. Note that on Windows systems, this must be the absolute path to the **`ldif2db.bat`** batch file and must include the "**`.bat`**" extension. This is a required parameter.

Database Name

This specifies the name of the database into which the LDIF data is to be imported. This is a required parameter, and it is case sensitive.

LDIF File

This specifies the absolute path to the LDIF file to be imported. This is a required parameter.

Log Command Output

This indicates whether the output of the `ldif2db` command should be captured and written as log messages.

LDAP CompRate

The LDAP CompRate job may be used to measure the performance of the directory server while performing LDAP compare operations. It is actually very similar to the LDAP ModRate job class, with the exception that instead of modifying a particular attribute within the entries that it targets, it will perform a comparison on the value of that attribute.

Because the LDAP CompRate job is so similar to the LDAP ModRate job, most of the parameters are identical. As such, they will not be repeated here. The only differences are parameters that the LDAP ModRate contains but are not needed by the LDAP CompRate job. Those are the "**Value Length**" parameter, the "**Modify Time Limit**" parameter, and the "**Follow Referrals**" parameter.

LDAP Load Generator

The LDAP Load Generator job provides a means of performing a number of different kinds of LDAP operations within the same job, using a weighting mechanism to control how frequently each type of operation is performed.

Note that even though this operation performs add, delete, and modify operations, it is possible to use across multiple clients, which is not possible with the other

SLAMD jobs that may be targeted at add and delete operations. However, because of the way that this job handles add, delete, and modify DN operations there are a few things to consider:

- Delete and modify DN operations will only be targeted at entries that have been added during the course of the job. As such, if you want to perform delete and/or modify DN operations, you must also perform add operations.

- If a delete or modify operation is chosen, but there are no entries remaining that have been added during the progress of the job, then an entry will be added instead. If this occurs, then it may cause the actual frequencies for add, delete, and modify DN operations to be different from what was originally requested.

- When the job is complete, there will likely be entries left in the directory that were added during processing but was never deleted. It is possible to perform cleanup after the job has completed to ensure that these entries are removed.

There are a number of parameters that may be configured when scheduling an LDAP Load Generator job:

Directory Server Address

This specifies the address of the directory server to target with the LDAP operations. It may be either an IP address or a resolvable fully-qualified domain name. It may also be the address of a load balancer or a directory proxy server, although it should be noted that the client must be allowed to perform the requested operations in the server or follow a referral to a server that will allow the operation.

Directory Server Port

This specifies the port number of the directory server to target with the LDAP operations. If the communication is to be performed over SSL, then this should be the secure port of the directory server.

Directory Base DN

This specifies the base DN to use when communicating with the directory server. This will be used as the search base for search operations, and it will also be the parent of any entries that may be added to the server while this job is active.

Bind DN

This specifies the user as whom the clients will bind to the server when issuing the requests. Note that this user must either have permission to perform all requested operations, or should have the ability to use the proxied authorization control.

Bind Password

This specifies the password to use for the bind DN.

Proxy As DN

This indicates whether the proxied authorization control should be used, and if so the DN of the user under whose authority the operations should be performed. If no value is provided, then the proxied authorization control will not be used and the user specified as the bind DN will be used to perform all operations. If a value is provided, then it should be the DN of the user under whose authority all operations should be performed, and the user specified as the bind DN should have permission to use the proxied authorization control.

Add Operation Frequency
Compare Operation Frequency
Delete Operation Frequency
Modify Operation Frequency
Modify RDN Operation Frequency
Search Operation Frequency

These parameters specify how frequently each of the associated types of operations should be performed. They must be integer values, and they should be relative frequencies (i.e., an operation with a frequency value of 2 should be performed approximately twice as often as an operation with a frequency value of 1). These may be percentage values if desired (i.e., the sum of all frequencies equals 100) as long as all values are integers. Any operation with a frequency value of zero will not be performed during this job, although at least operation must have a nonzero frequency.

DN File URL

This specifies the URL to a file containing the DNs of the entries to target with compare and modify operations. It is an alternative to providing an entry DN or DN pattern, and if it is provided, then it should be a URL that can reference a file on a remote server using an HTTP URL like:

```
http://server.example.com/dns.txt
https://server.example.com/dns.txt
ftp://server.example.com/data/dns.txt
http://user:password@server.example.com/dns.txt
https://user:password@server.example.com/dns.txt
ftp://user:password@server.example.com/data/dns.txt
```

or on the local filesystem with a URL like:

```
file:/data/dns.txt
```

Note that if a DN file is used, it should contain one entry DN per line. Whenever a modify or compare operation is to be performed, a DN will be chosen by randomly selecting a different line from the file.

Entry DN

This specifies the DN of the entry that should be targeted by compare and modify operations. It is an alternative to providing a DN file URL, and it can specify a range of entries through the use of a bracketed numeric range. For example, a value of

```
uid=user.[1-1000000],ou=People,dc=example,dc=com
```

will cause an integer value between 1 and 1000000 (inclusive) to be chosen at random and used in place of the bracketed range. Alternatively, you can replace the dash with a colon, like

```
uid=user.[1:1000000],ou=People,dc=example,dc=com
```

and the values will be chosen sequentially from 1 to 1000000 rather than at random. Note that the sequential ordering will not be maintained across multiple clients, although it can work across multiple threads in the same client.

Search Filter File URL

This specifies the URL to a file that contains the filters to use when performing search operations in the directory server. This is an alternative to providing a single search filter or filter pattern, and if it is specified then it should represent the URL to a file containing a set of search filters, with one filter per line. It may specify a file on a remote server using the HTTP protocol, like:

```
http://server.example.com/filters.txt
https://server.example.com/filters.txt
ftp://server.example.com/data/filters.txt
http://user:password@server.example.com/filters.txt
https://user:password@server.example.com/filters.txt
ftp://user:password@server.example.com/data/filters.txt
```

or a file on the local filesystem using a FILE URL, like:

```
file:/data/filters.txt
```

Search Filter

This specifies the filter or filter pattern to use when performing search operations. It may be a single search filter, but it may also target multiple entries by providing a bracketed numeric range. For example, a filter of:

```
(uid=user.[1-1000000])
```

will cause the client to choose a number at random between 1 and 1000000 (inclusive) and use that number in place of the bracketed range. Similarly, using a colon in place of a dash, like:

```
(uid=user.[1:1000000])
```

will use a sequentially-incrementing number in the search filters, starting at 1 and increasing up to 1000000. After all values have been exhausted, it will start back at 1. Note that the sequential ordering can be maintained across multiple threads in the same client, but it cannot be enforced across separate clients.

Attribute to Compare/Modify

This specifies the name of the attribute that will be targeted for compare and modify operations.

Search Size Limit

This specifies the maximum number of entries that should be returned from any single search operation. The directory server should only return up to this number of entries per search, even if the search criteria match more than this number of entries in the server.

Operation Time Limit

This specifies the maximum length of time in seconds that any operation should be allowed to take. If an operation takes longer than this length of time to complete, then it will be abandoned and the next request will be sent.

Warm Up Time

This specifies the length of time in seconds that the job will be allowed to run before the client starts to collect statistics. This makes it possible to ensure that all clients and threads have started running and are generating a consistent load against the server before starting to collect statistics. Note that this is considered part of the overall job duration, so it may be desirable to increase the overall job duration to account for this time.

Cool Down Time

This specifies the length of time in seconds before the job ends that the clients should stop collecting statistics. This makes it possible to ensure that the load on the server is still consistent at the time that statistics collection ends. Note that this is considered part of the overall job duration, so it may be desirable to increase the duration to account for this time.

Time Between Requests

This specifies the length of time in milliseconds between requests sent to the server by each client thread. Note that this time is measured between the beginning of one request and the beginning of the next, rather than between the end of one request and the beginning of the next. This makes it possible to target a particular number of operations per second by adjusting the number of clients, threads per client, and delay between requests. Note that if any operation takes longer than this length of time to complete, there will be no delay before sending the next request.

Use SSL

This indicates whether SSL will be used to encrypt the communication between the clients and the server.

Blindly Trust Any Certificate

This indicates whether the clients should blindly trust any SSL certificate presented by the directory server. If this is not checked, then the client will verify the certificate presented by the server using the specified SSL trust store.

SSL Key Store

This specifies the location of the JSSE key store that will be used if the directory server is configured to require SSL client authentication. Note that even if the client does present a certificate to the directory server, that certificate will not be used for LDAP authentication as this job does not support the SASL EXTERNAL mechanism.

SSL Key Store Password

This specifies the password that should be used to access the information in the JSSE key store should the client need to present a certificate to the directory server.

SSL Trust Store

This specifies the location of the JSSE trust store that will be used to verify the SSL certificate presented by the directory server, provided that the job has not been configured to blindly trust any certificate.

SSL Trust Store Password

This specifies the password that will be used to access the information in the SSL trust store, which should only be required if the SSL trust store is needed and is in a format that requires a password for read-only access.

Clean Up When Done

This indicates whether the clients should perform any cleanup that may be needed after job processing has completed. If so, then any entries added but not deleted while the job was running will be removed.

Always Disconnect

This indicates whether a separate connection should be used for each request sent to the server. By default, each thread will establish a connection to the server at the beginning of the job and will use that connection for all operations performed while the job is active. If this is checked, however, then a new connection will be established before each request and disconnected after receiving the response.

Follow Referrals

This indicates whether clients should follow any referrals that they may receive while sending requests to the server. If this is not checked, then no referrals will be followed.

LDAP Weighted Load Generator

The LDAP Weighted Load Generator job operates in much the same way as the LDAP Load Generator job, with the exception that it can use a weighted access pattern for targeting entries. There differences between the parameters for this job and the LDAP Load Generator job are:

- It is not possible to retrieve the entry DNs or search filters from a file. Rather, the DNs and filters to use must be provided as patterns.
- The "**Entry DN**" parameter has been split into "**Entry DN 1**" and "**Entry DN 2**" parameters. Both parameters accept a pattern including a bracketed numeric range that can be used to target multiple entries.
- The new "**DN 1 Percentage**" parameter specifies the percentage of the time that the pattern specified in the "**Entry DN 1**" parameter will be used to generate the DN of the entry to target for the corresponding operation. It must be an integer value between 0 and 100, inclusive.
- The "**Search Filter**" parameter has been split into "**Search Filter 1**" and "**Search Filter 2**" parameters. Both parameters accept a pattern including a bracketed numeric range that can be used to target multiple entries.
- The new "**Filter 1 Percentage**" parameter specifies the percentage of the time that the pattern specified in the "**Search Filter 1**" parameter will be used to choose the filter to use for a search operation. It must be an integer value between 0 and 100, inclusive.

LDAP Load Generator with Replication Latency

The LDAP Generator with Replication Latency job provides the same set of functionality as the LDAP Load Generator job, with the added benefit of being

able to measure the time required for changes to replicate from one directory (the master) to another (the replica). The parameters for this job differ from those of the LDAP Load Generator job in the following ways:

- This job does not offer the ability to use SSL.
- The "**Directory Server Address**" and "**Directory Server Port**" parameters have been renamed to "**Master Directory Host**" and "**Master Directory Port**", respectively. These specify the information used to connect to the master directory, in which the changes will be made.
- New "**Replica Directory Host**" and "**Replica Directory Port**" parameters have been added that can be used to provide the information needed to connect to the replica directory server that will be used to detect the changes made in the master.
- A new "**Latency Check Entry DN**" parameter has been added that can be used to specify the DN of the entry that should be periodically modified on the master, and against which the persistent search will be registered on the replica server. Note that this entry should be one that is not modified as part of the normal load generation process, nor should it be modified by anything other than the latency checking process itself. Any updates to this entry not performed by the latency checking process will interfere with the accuracy of the latency check measurements.
- A new "**Time Between Latency Checks (ms)**" parameter has been added that specifies the minimum length of time in milliseconds to allow between updates to the latency check entry. If it takes less time than specified by this parameter for a change to be propagated from the master to the replica, then the latency checking process will sleep for the remainder of that time before performing the next modification against the latency check entry. If it takes more than this time for the change to be replicated, then there will be no delay before the next update attempt.

LDAP Load Generator (with multiple searches)

The LDAP Load Generator job can be very useful for performing a mix of LDAP operations against the server, but one shortcoming that it has is that it lumps all kinds of search operations (presence, equality, substring, approximate, etc.) into one category. While it is possible to place multiple kinds of search operations in the same filter file, it is difficult to control the relative frequencies of each kind of filter. This is addressed by the LDAP Load Generator (with multiple searches)

job because it is possible to specify up to six filter files with different weights for each.

The other main difference between this job and the LDAP Load Generator job is the way in which binds are handled. Rather than use a single bind DN for the duration of the job, the LDAP Load Generator (with multiple searches) makes it possible to bind as multiple users through the use of a bind DN pattern. For example, specifying a bind DN of:

```
uid=user.[1-1000000],ou=People,dc=example,dc=com
```

will cause an integer value to be chosen between 1 and 1000000 (inclusive) and used in place of the bracketed range. Similarly, using a colon, like:

```
uid=user.[1:1000000],ou=People,dc=example,dc=com
```

will cause the values to be chosen in sequential order rather than at random.

Not only will there be a bind as part of establishing the initial connection, but it is also possible to bind periodically throughout the job. If a positive integer value is specified for the "**Operations Between Binds**" parameter, then each thread will choose a new bind DN and perform a bind as that user after that number of operations have been performed. A value of zero indicates that no rebinds should occur during the course of the job processing.

Because of the change in the bind behavior associated with this job, there are two other changes in the LDAP Load Generator (with multiple searches) job as compared to the LDAP Load Generator job. The first is that unlike the LDAP Load Generator job, this job does not provide the ability to use the proxied authorization control and therefore, all users specified must have permission to perform the requested operations. The second change is that rather than specifying an entry DN pattern or a file containing a list of DNs, compare and modify operations are targeted at the entry for the currently authenticated user.

LDAP Search and Modify Load Generator

The LDAP Search and Modify Load Generator job combines search and modify operations in the same job. It is essentially the same as the LDAP SearchRate job, with the exception that it will also modify every entry returned from the searches. Because this job is so similar to the LDAP SearchRate job, only the differences in the parameters will be discussed here:

- The search base associated with the LDAP SearchRate job can be specified as a DN pattern with a bracketed numeric range to use a number of different search bases. With this job, only one search base may be specified. In addition, all searches performed will have a subtree scope.

- The LDAP Search and Modify Load Generator job introduces a new configuration parameter, "**Attributes to Modify**", in which it is possible to specify one or more attributes that should be modified in each entry returned from the search. Multiple attributes can be specified by placing the names of the attributes on separate lines.

- The LDAP Search and Modify Load Generator job does not offer the capability to disconnect after each operation or set of operations.

LDAP Weighted Search and Modify Load Generator

The LDAP Weighted Search and Modify Load Generator job is nearly the same as the LDAP Search and Modify Load Generator job, with the exception that it adds weighting capabilities so that different sets of entries can be targeted with different frequencies. In particular, rather than providing a file containing search filters or a single search filter or filter pattern, this job makes use of two search filter patterns and allows the user to determine how frequently each is used to perform a search. All parameters associated with the LDAP Weighted Search and Modify Load Generator job are discussed earlier in this document and will not be restated here.

The Mail Jobs

Although many of the jobs provided with SLAMD are targeted at LDAP directory servers, it is equally well suited for other kinds of network applications. This section will focus on the jobs that make it possible to apply load against an e-mail server, by communicating over IMAP, POP, and SMTP.

IMAP CheckRate

The IMAP CheckRate job generates load against an IMAP mail server by establishing a connection, authenticating to the server, selecting the INBOX folder, retrieving a list of all the messages, and logging out and disconnecting. Note that the process of retrieving the list of messages is done using the IMAP **FETCH** command with the **(FLAGS)** qualifier, which only retrieves a summary of the messages rather than the entire body.

The parameters that may be specified when scheduling an IMAP CheckRate job are as follows:

IMAP Server Address

This specifies the address that clients should use when communicating with the IMAP server. It may be an IP address or a resolvable fully-qualified name, and it may also be a load-balanced address.

IMAP Server Port

This specifies the port number to use when communicating with the IMAP server. The default port number is 143, but any port may be used provided that an IMAP server is listening on that port.

User ID

This specifies the user ID to use when authenticating to the IMAP server. It may be a single user ID, but it may also specify a range of entries by enclosing that range in brackets. For example, a user ID of

```
user.[1-1000000]
```

will cause the client to choose an integer value between 1 and 1000000 (inclusive) and use that number in place of the bracketed range. Similarly, replacing the dash with a colon like

```
user.[1:1000000]
```

will cause the client to choose the values sequentially rather than at random.

User Password

This specifies the password to use when authenticating to the IMAP server. Note that if a user ID range is to be used, then all users should have the same password for the authentications to succeed.

Time Between IMAP Sessions

This specifies the length of time in milliseconds that should be allowed to pass between IMAP sessions. Note that this time is measured from the beginning of one session to the beginning of the next, making it possible to target a specific number of IMAP sessions per second. However, if it takes longer than this length of time for any IMAP session to complete, then there will be no delay before starting the next.

POP CheckRate

The POP CheckRate job performs essentially the same task as the IMAP CheckRate job, with the exception that it communicates using the POP3 protocol rather than IMAP. That is, it establishes a connection to the POP server, authenticates, retrieves a list of messages, and disconnects. The process of retrieving the list of messages is performed using the POP LIST command, so only the message numbers and sizes are retrieved, not the entire message bodies.

Because the work performed by the POP CheckRate job is identical to that of the IMAP CheckRate job but with a different protocol, the parameters that may be specified for both jobs are also identical. As such, they will not be repeated here.

SMTP SendRate

The SMTP SendRate job performs the task of sending e-mail messages of a specified size to one or more recipients using the SMTP protocol. In particular, the job will establish a connection to the SMTP server, specify the set of recipients, provide the message, and disconnect from the server.

The set of parameters that may be specified when scheduling an SMTP SendRate job include:

SMTP Server Address

This specifies the address that clients should use when communicating with the SMTP server. It may be either an IP address or a fully-qualified name.

SMTP Server Port

This specifies the port number that clients should use to connect to the SMTP server. The standard SMTP port number is 25, but any port may be used if the client can communicate on that port using SMTP.

From Address

This specifies the address from which the e-mail message should originate. Although this job will not require that it be a real address, the mail server configuration may require that it be valid.

Recipient Address

This specifies the address or address pattern that should be used when determining the recipient(s) to use when sending messages. This may be a single address, but it may also include a bracketed numeric range. For example, the address

```
user.[1-1000000]@example.com
```

will cause the client to choose an integer value between 1 and 1000000 at random and use it in place of the numeric range. Similarly, the address

```
user.[1:1000000]@example.com
```

will cause the values to be chosen in sequential order rather than at random. Note that the sequential ordering can be maintained across multiple threads on the same client, but it will not be separated across multiple clients.

Minimum Number of Recipients
Maximum Number of Recipients

This specifies the minimum and maximum number of recipients that should be used in the e-mail messages to send. A random number will be chosen between the minimum and maximum values (inclusive) for each message to send.

Message Body Size

This specifies the size in bytes to use for the messages to be sent. Note that this will only be an approximation, and in many cases the message will be slightly larger than this size. Further, this size does not include the SMTP header that will be added to the messages.

Time Between SMTP Sessions

This specifies the length of time in milliseconds that will be allowed to pass between each SMTP session within a client thread. Note that this time is measured between the beginning of one SMTP session and the beginning of the next, rather than the end of one session and the end of the next. This will make it possible to target a specific number of SMTP sessions per second by varying the number of clients, threads per client, and time between SMTP sessions. However, if any SMTP session takes longer than this length of time to complete, then there will be no delay before the next request.

The HTTP Jobs

SLAMD offers basic capabilities for communicating with Web servers using the HTTP protocol. While the jobs provided by default may not be suitable for all cases, they may be acceptable for general load testing, and they demonstrate that the API that SLAMD provides for developing custom jobs may be used for HTTP just as it can be for other protocols like LDAP, IMAP, POP, and SMTP.

HTTP GetRate

The HTTP GetRate job may be used to repeatedly perform HTTP GET operations against one or more Web servers. It provides a number of options to customize the way that the requests are performed, including the use of SSL, performing operations through a proxy server, automatically following redirects, and automatically downloading files associated with HTML documents that are retrieved.

The parameters that may be specified with the HTTP GetRate are as follows:

URL To Retrieve

This specifies the URL to the file that should be retrieved. It should be in one of the following forms:

`http://`*{host}*`/`*{file}*
`http://`*{host}*`:`*{port}*`/`*{file}*
`https://`*{host}*`/`*{file}*
`https://`*{host}*`:`*{port}*`/`*{file}*

where *{host}* is the address of the Web server from which the file should be retrieved, and it should be either an IP address or resolvable name. The *{file}* component specifies the location of the file on that server relative to the document root. An optional *{port}* may be specified as well, although if no port is provided then the default port number will be used (80 for HTTP connections, or 443 for HTTPS connections).

File with URLs to Retrieve

This specifies the URL to a file containing the sets of URLs to use in the requests generated. This provides an alternative to using a single URL. If this is used, then a separate URL will be chosen at random for each request.

Use KeepAlive

This indicates whether the HTTP KeepAlive option should be used, which will allow a single connection to be used for multiple requests to the same server, rather than establishing a separate connection for each request.

Follow Redirects

This indicates whether the client should automatically follow any HTTP redirects that it encounters while processing a request. If this feature is enabled, then the client will automatically retrieve the URL specified by the redirect. Otherwise, the client will report the redirect itself in the statistics retrieved.

Retrieve Associated Files

This indicates whether the client should automatically try to download any associated files when retrieving an HTML document. If this is enabled, and if the response from the HTTP server is an HTML document, then the client will parse that HTML document and extract URLs to embedded images as well as external stylesheets, frames, and scripts that it may contain. It will then attempt to retrieve those files as part of processing the request. This allows the client to more closely simulate the behavior of an actual browser.

Proxy Server Address

This specifies the address that should be used for the HTTP proxy server through which the requests should be sent. If no value is provided for this parameter, then no proxy server will be used.

Proxy Server Port

This specifies the port that should be used for the HTTP proxy server through which the requests should be sent. If no value is provided for this parameter, then no proxy server will be used.

Warm Up Time

This specifies the length of time in seconds that the job should be allowed to run before starting to collect statistics. This makes it possible to ensure that all client threads have started and that there is a consistent load on the server before starting to collect statistics. Note that this is included as part of the overall job duration, and therefore if a warm up time is used then it may be desirable to increase the job duration to account for the warm up time.

Cool Down Time

This specifies the length of time in seconds before the job completes running that it should stop collecting statistics. This makes it possible to ensure that the load on the server is still consistent when the statistics collection ends. This length of time is included as part of the overall job duration, and therefore if a cool down time is used then it may be desirable to increase the job duration accordingly. Also note that the cool down time will only be used if the job ends due to the stop time or maximum duration. It will not be used if the job ends after completing the maximum number of iterations.

Number of Iterations

This specifies the maximum number of requests that each thread will send to the Web server. After this number of requests have been sent, the thread will complete, and after all thread have completed, the job will end.

Time Between Requests

This specifies the length of time in milliseconds that should be allowed between requests to the Web server. Note that this time is measured between the beginning of one request to the beginning of the next, which makes it possible to target a specific number of operations per second by adjusting the number of clients, threads per client, and delay between requests.

Blindly Trust Any Certificate

This indicates whether the clients should blindly trust any SSL certificate presented by the Web server. If this is not checked, then clients will use the information in the SSL trust store to determine whether to trust the certificate provided by the server.

SSL Key Store

This specifies the location of the JSSE key store that should be used if the client needs to present a certificate to the Web server. Whether or not this certificate will be used for authentication is dependent upon the Web server configuration.

SSL Key Store Password

This specifies the password that will be used to access the information in the JSSE key store if it should be required.

SSL Trust Store

This specifies the location of the JSSE trust store that will be used if the client needs to verify the certificate presented by the Web server rather than blindly trusting any certificate.

SSL Trust Store Password

This specifies the password needed to access the information in the JSSE trust store. This should only be needed if the client needs to verify the certificate presented by the Web server and the SSL trust store is in a format that requires a password for read-only access.

Calendar Initial Page Rate

The Calendar Initial Page Rate job is a job specifically designed to measure the performance of the Sun ONE Calendar Server version 5.1.1 when authenticating users and displaying the initial calendar page. Note that this job is specifically designed to work with the Sun ONE Calendar Server 5.1.1 and will likely not work with any other software. However, it does demonstrate that SLAMD may be used to drive HTTP-based applications.

The following parameters may be specified when scheduling a Calendar Initial Page Rate job:

Calendar Server Address

This specifies the address that the clients should use when communicating with the calendar server. It may be either an IP address or a fully-qualified resolvable name.

Calendar Server Port

This specifies the port that clients should use to connect to the calendar server. By default, the calendar server will listen on port 80, but it may be reconfigured to listen on any port.

User ID

This specifies the user ID or user ID pattern that should be used when authenticating to the calendar server. It may be a single user ID, but it may also contain a bracketed numeric range to specify a range of users. For example, a user ID of:

```
user.[1-1000000]
```

will cause the client to choose a number at random between 1 and 1000000 and use that number in place of the bracketed range. In addition, it is possible to use a colon instead of a dash, like:

```
user.[1:1000000]
```

to cause the numbers to be chosen sequentially rather than at random. Note, however, that while this sequential ordering can be maintained across multiple threads in the same client, it cannot be preserved across multiple clients.

User Password

This specifies the password that should be used when authenticating to the calendar server. Note that if a user ID pattern has been provided to target multiple users, all users should have the same password.

Time Between Login Attempts

This specifies the length of time in milliseconds that should be allowed between calendar sessions. Note that this time is measured from the beginning of one session to the beginning of the next, which makes it possible to target a specific number of operations per second by adjusting the number of clients, threads per

client, and delay between sessions. If any session takes longer than this length of time to complete, then there will be no delay before starting the next session.

The Identity Server Jobs

The Identity Server jobs provided with SLAMD may be used to interact with the Sun ONE Identity Server over HTTP. These jobs should work with both Identity Server version 6.0 and version 6.1.

Identity Server Authentication Rate

The Identity Server Authentication Rate job provides a means for measuring the pure rate of authentication sustainable by the Sun ONE Identity Server. The job itself performs what Identity Server refers to as a "0 page login". It directly posts credentials to the Authentication Service, and reads the result back from the server. Success or failure of the authentication attempt is recorded, along with the number of attempts and the time that the authentication took. This allows for measuring the throughput (number of attempts per interval) and the latency (time per authentication attempt) of the authentication system in isolation under various loads generated by SLAMD.

The configuration parameters that may be specified when scheduling an Identity Server Authentication Rate job include:

Authentication Service URL

This specifies the URL to which credentials will be directly posted. It should be a valid URL for authentication to the Identity Server. The default Authentication URL is

`http://`*{address}*`:`*{port}*`/amserver/UI/Login`**

for non-secure communication, or

`https://`*{address}*`:`*{port}*`/amserver/UI/Login`**

for encryption using SSL. It may also include advanced login parameters by appending them to the URL as a query string (e.g., "`?module=Unix`"). Consult the Identity Server documentation for the available parameters.

Protocol Version

This indicates the HTTP protocol version that should be used when communicating with the Identity Server. This job provides support for both HTTP 1.0 and HTTP 1.1.

Use KeepAlive

This indicates whether connections using HTTP 1.1 should use the KeepAlive option to allow a single connection to be used for multiple requests.

Login Data File URL

This specifies the URL to a file containing the information to use in the authentication process. This is an alternative to providing a single login ID value or value pattern. If a file is provided, then each line should contain information about one user, with the login ID at the beginning of the line followed by a tab and the password for that user.

If a login data file is provided, it should be given as a URL. The URL may reference a file on a remote server using HTTP, like:

```
http://server.example.com/userinfo.txt
```

or on the local filesystem, like:

```
file:/data/authinfo.txt
```

Login ID Value

This specifies the login ID that should be used when authenticating to the Identity Server, and it is an alternative to providing this information in a login data file. It may be a single login ID, but it may also contain a bracketed numeric range to indicate a number of users. For example,

```
user.[1-1000000]
```

indicates that a random integer between 1 and 1000000 (inclusive) should be chosen and used in place of that bracketed range. Similarly, if a colon is used in place of the dash, like:

```
user.[1:1000000]
```

then the client will increment through through the values sequentially. Note that the process of incrementing these values sequentially can be properly maintained across all the threads on a single client, but because there is no communication between clients it is not possible to maintain sequential ordering across multiple clients.

Login Password

This specifies the password that should be used when authenticating to the Identity Server. All users are expected to have the same password when using this option. If all users do not have the same password, then it will be necessary to use the login data file URL option to provide a file with the appropriate information for each user.

Login ID Token Name

This specifies the name of the form parameter which will hold the user identifier portion of the user's credentials. The default value of "Login.Token1" will work with both Identity Server 6.0 and 6.1 and should not need modification.

Password Token Name

This specifies the name of the form parameter which will hold the password (or other credential) portion of the user's credentials. The default value of "Login.Token2" will work with both Identity Server 6.0 and 6.1 and should not need modification.

Warm Up Time

This specifies the length of time in seconds that the job should be allowed to run before starting to collect statistics. This makes it possible to ensure that the all clients have started processing and that the load against the server has stabilized before beginning statistics collection.

Cool Down Time

This specifies the length of time in seconds before the job completes that it should stop collecting statistics. This makes it possible to ensure that the load against the server is still consistent when statistics collection ends. Note, however, that the

cool down time may only be effectively used if the job stops running as a result of reaching the stop time or running for the maximum allowed duration. If the job completes for any other reason (e.g., completing the specified number of iterations), then this cool down time will not be used.

Number of Iterations

This specifies the maximum number of authentications that each thread should perform against the Identity Server. The total number of authentication attempts will be equal to this value multiplied by the number of clients used to run the job multiplied by the number of threads per client. The default value of -1 indicates that there should not be a maximum number of iterations.

Time Between Requests

This specifies the length of time in milliseconds that each thread should allow when sending authentication requests to the Identity Server. Note that this time is measured from the beginning of one request to the beginning of the next (as opposed to the end of one request to the beginning of the next), which makes it possible to target a specific rate of operations. Note that if any single authentication takes longer than this length of time, then there will be no delay before sending the next request.

Blindly Trust Any Certificate

This indicates whether the job should blindly trust any certificate that may be presented by the Identity Server if authentication is performed using SSL. If this option is used, then it will not be necessary to ensure that the server certificate is trusted through another means (e.g., by including the issuer's certificate in the SSL trust store).

SSL Key Store

This specifies the location of the JSSE key store that the client should use when communicating with the Identity Server over SSL. Note that a JSSE key store will only be required if the server is configured to require clients to present a certificate as part of the authentication process.

SSL Key Store Password

This specifies the password that should be used to access the information contained in the JSSE keystore.

SSL Trust Store

This specifies the location of the JSSE trust store that the client should use to determine whether the SSL certificate presented by the Identity Server is valid. This should only be necessary if the authentication process is used over SSL and the job is not configured to blindly trust any certificate.

SSL Trust Store Password

This specifies the password that should be used to access the information contained in the JSSE trust store. Unless the trust store is in a special format that requires a password in order to access the information used for certificate validation, no password should be required.

Disable SSL Session Caching

This indicates whether the client should manually invalidate SSL sessions upon establishing connections to the Identity Server. SSL session caching may be used to reduce the amount of work required to establish further SSL connections, but it does not accurately simulate connections coming from multiple client systems.

Identity Server Benchmark Client

The Identity Server Benchmark Client job is one which can be used for simulating a wide variety of load against the Identity Server. This makes it possible to integrate agents and perform authentication, authorization checks, and session logout operations, which provides the ability to mimic a variety of use cases. The job is flexible in configuration, and can be used as simply as the Identity Server Authentication Rate job class, or as complex as the Mindcraft benchmark. The job will measure total transaction time, as well as data on each of the subcomponents of a transaction. Each of these are counted an timed, and their result codes are also tracked.

It should be noted that this client does not currently load supporting files in an HTML page (e.g., images, javascript files, CSS files, frame targets, etc.). This should be considered when evaluating overall performance results.

The configuration parameters that may be specified when scheduling an Identity Server Benchmark Client job are:

Un-Authenticated URLs to Load

This specifies the list of URLs that will be loaded by the client before any authentication is performed. This provides the flexibility to load a number of URLs before ever contacting an agent or the Identity Server. It typically might not be used in a benchmark, but it provides flexibility for the client to adapt to different test scenarios. The job will simply load each of these URLs before moving on to portions of the transaction specific to Identity Server. Timing of this is tracked independently, although the time will be the aggregate for loading all URLs in the list.

Use This Pre-Authentication URL

This specifies a URL that will be loaded as the initial part of an authentication transaction. This URL must result in the client loading the desired authentication URL. For example, it might be the URL of an agent-protected resource. This would result in the agent redirecting the user to an authentication service. The result of that redirection will be loaded, and that URL will be used as the Target of the authentication form after posting the user's credentials.

This URL may also be that of the authentication service itself, which would force the client to load the authentication service URL, effectively benchmarking the time required to load the form presented by the authentication service. If this URL is specified, then the "Skip Pre-Auth and Use this Authentication URL" parameter will not be used and the last URL loaded as a result of following the redirects will be used as the authentication URL. If no value is provided for the "Skip Pre-Auth and Use This Authentication URL" parameter, then this parameter must be given a value.

Skip Pre-Auth and Use This Authentication URL

This specifies the URL of the Identity Server authentication service. It allows the job to perform a "0 page login", in which the user's credentials are directly posted to the Identity Server without first loading the login form from the server. If no value is provided for the "Use This Pre-Authentication URL" parameter, then this parameter must be given a value.

Follow the Re-Direct Returned from Auth

This indicates whether the client will load the URL specified in the location header that is returned from the authentication service. By default, this would load the amconsole service, although it might also load a URL specified by a goto parameter. For instance, if an agent-protected pre-auth URL were specified, the

agent would redirect the client to the authentication service with a parameter of "**goto=***{user_requested_url}*". The redirect from the authentication service would then be to that requested URL. If this option is checked, then this URL will be loaded. If it is not checked, then the redirect URL will be ignored. This URL load will be tracked as part of the authorization count and timing.

Authorization URLs to Load

This specifies a list of URLs to load after performing authentication. This makes it possible to specify a number of agent-protected URLs that will be loaded. Since the SSO token will be presented to these URLs it may be used to benchmark agents interacting with the Identity Server. It may also be used for arbitrary purposes.

Sequential AuthZ URL

This specifies a URL pattern to use to construct a set of authorization URLs to be loaded after authentication. It may contain a bracketed numeric range, like:

```
http://server.example.com/[1-5].html
```

which will cause the job to iterate sequentially through the set of authorization URLs

```
http://server.example.com/1.html
http://server.example.com/2.html
http://server.example.com/3.html
http://server.example.com/4.html
http://server.example.com/5.html
```

If both a set of authorization URLs to load and a sequential authorization URL are specified, then the set of sequential URLs indicated by this value will be appended to the list of authorization URLs to load.

Time to Pause Between Authorizations

This specifies a delay in milliseconds that may be inserted between loading the individual authorization URLs. If a positive value is specified for this parameter, then the job will sleep for that number of milliseconds before loading each of the authorization URLs. A value of zero indicates that no delay should be used.

URL for Logout

This specifies a URL that may be loaded to terminate the session created during authentication. The default logout URL for Identity Server is

http://*{address}*:*{port}*/amserver/UI/Logout**

for unsecured communication, or

https://*{address}*:*{port}*/amserver/UI/Logout**

if SSL is to be used. If no logout URL is specified, the SLAMD client will discard the SSO token, but the session will remain on the Identity Server until it reaches the configured session timeout.

Load Entire Page

This indicates whether the client should read all the information from an incoming input stream. If this is not checked, then only the HTTP header of the response will be read in order to obtain the information required for further processing, but the actual data will not be read. If only the header information is read, then the client may be free to place more load against the server.

Login Data File URL

This specifies the URL to a file containing information to use in the authentication process. This is an alternative to providing a single login ID value or value pattern and password, and if specified should be either an HTTP URL to load the data file from a remote server, like:

http://server.example.com/userinfo.txt

or from the local filesystem, like:

file:/data/userinfo.txt

If a login data file is provided, then it should contain information about one user per line, with the login ID followed by a tab and the corresponding password.

Login ID Value

This specifies the login ID to use when authenticating to the Identity Server. It may be a single login ID, but it may also specify a pattern of IDs to use when authenticating by including a bracketed numeric range. For example, a login ID value of:

user.[1-1000000]

will cause the client to choose a value at random between 1 and 1000000 (inclusive) to use in place of the bracketed range. Similarly, using a colon instead of a dash like:

```
user.[1:1000000]
```

will cause the client to iterate sequentially through the values rather than choosing them at random.

Login Password

This specifies the password to use when authenticating to the Identity Server. If a login ID or login ID pattern is to be used rather than providing the information in a data file URL, then all users must have the same password for authentication to succeed.

Warm Up Time

This specifies the length of time in seconds that the job should be allowed to run before it begins to collect statistics. This makes it possible to ensure that all clients have started and that the load on the server is consistent before beginning to collect statistics.

Cool Down Time

This specifies the length of time in seconds before the job ends that it should stop collecting statistics. This helps make it possible to ensure that the load against the server is still stable when statistics collection ends. Note that this may only be used accurately if the job ends as a result of reaching the scheduled stop time or running for the specified maximum duration.

Number of Iterations

This specifies the number of iterations that each thread should perform before the job completes. A value of -1 indicates that there is no limit and that the job should run until it reaches the stop time or maximum duration.

Time Between Requests

This specifies the minimum length of time that should be allowed to pass between the beginning of one iteration and the beginning of the next. This makes it

possible to target a specific rate of operations against the server. A value of zero indicates that there should not be any delay between iterations.

SSL Is Being Used

This indicates whether the client will perform any communication over SSL during the course of its processing. If SSL is to be used, then this option should be checked so that the appropriate initialization will be performed.

Note that at the present time, the Identity Server Benchmark Client job only supports scenarios in which all communication is protected using SSL, or cases in which all communication is unencrypted. It is not currently possible to retrieve some URLs using HTTP and others using HTTPS.

Blindly Trust Any Certificate

This indicates whether the client should blindly trust any SSL certificate that may be presented to it. This can simplify the process of establishing SSL-based connections to servers without the need to update all the clients with the information necessary to trust the server certificates.

SSL Key Store

This specifies the location of the JSSE trust store to use when establishing SSL-based connections to the server. This should only be needed if the server is configured to require clients to present their own certificate in order to establish a connection.

SSL Key Store Password

This specifies the password that should be used to access the information in the JSSE key store. This should not be required unless a JSSE key store is needed for the client to present its own certificate to the server.

SSL Trust Store

This specifies the location of the JSSE trust store that should be used to determine whether to trust the SSL certificate presented by the server. This should only be needed if the client needs to communicate over SSL and is not configured to blindly trust any certificate that may be presented.

SSL Trust Store Password

This specifies the password that should be used to access the information in the JSSE trust store. Most trust store formats do not require a password to retrieve information used to validate a server certificate, so this should not be required in most cases.

Disable SSL Session Caching

This indicates whether to disable SSL session caching on the client side. This may result in lower performance because it will require the clients to do more work when establishing SSL-based connections to the server, but it may also provide a more accurate simulation of real-world performance that can be expected when connections are coming from a large number of client systems.

The SQL Jobs

SLAMD offers a number of jobs that can be used to help test the search and modify performance of LDAP directory servers. However, it also offers jobs for performing similar tests against relational databases using SQL. These include the SQL SearchRate and SQL ModRate jobs.

Note that the jobs that SLAMD provides for communicating with relational databases do so using Java's JDBC API. The jobs can be used to communicate with any relational database for which a JDBC driver exists (including Oracle, DB2, MySQL, and many others), but it is important to note that SLAMD itself does not provide any JDBC drivers. Therefore, before being able to use these jobs, it is necessary to install the appropriate driver on all clients that will be used to run them. If the driver is provided in a JAR file, then that JAR file may be placed in the `lib` directory under the client installation. If the driver is provided as one or more individual Java classes, then the class(es) may be placed under the `classes` directory structure, using the appropriate directory hierarchy based on the package in which those classes are defined.

SQL SearchRate

The SQL SearchRate job is intended to measure the query performance of relational databases. It does so by repeatedly issuing queries against the database and measuring the number of searches performed per second, the average duration of each search in milliseconds, and the average number of rows returned per search.

The following parameters may be specified when scheduling an SQL SearchRate job:

JDBC Driver Class

This specifies the fully-qualified class name for the JDBC driver that the clients should use when communicating with the database.

JDBC URL

This specifies the URL that should be used to connect to the database. It is typically in a form like "**jdbc:***{dbtype}***:***{driverdata}*", where "*{dbtype}*" is the database type being used and "*{driverdata}*" is is a string that provides the information necessary for connecting to the database. The value of the "*{dbtype}*" component, as well as the format of the "*{driverdata}*" portion are both specific to the JDBC driver being used.

User Name

This specifies the username of the account to use when accessing the database. It may be left blank if no authentication should be performed.

User Password

This specifies the password to use when accessing the database. It may be left blank if no authentication is to be performed, or if the account used does not have a password.

SQL Query

This specifies the SQL query to issue to the database. It may be a single query, but it may also contain a bracketed numeric range to alter the query for each search. For example, using a query of

```
SELECT email FROM users WHERE uid = user.[1-1000000]
```

would cause the client to choose a random number between 1 and 1000000 and use it in place of the bracketed range for each query. Similarly, replacing the dash with a colon, like

```
SELECT email FROM users WHERE uid = user.[1:1000000]
```

would cause the values to be chosen sequentially rather than at random.

Warm Up Time

This specifies the length of time in seconds that the job should be allowed to run before starting to collect statistics. This can be used to help ensure that all clients have connected to the database and that the load is stable before beginning to measure statistics.

Cool Down Time

This specifies the length of time in seconds before the job ends that it should stop collecting statistics. This can be used to help ensure that all clients are still connected and generating a consistent load against the database when ending statistics collection. Note that this will only be available if the job ends because the stop time or maximum duration is reached. It will not be used if the job ends because it has completed the specified number of iterations or if it is cancelled by an administrator.

Time Between Queries

This specifies the length of time in milliseconds that each thread should allow between queries sent to the database. Note that this time is measured from the beginning of one query to the beginning of the next, making it possible to control the rate at which queries are issued to the database.

Number of Iterations

This specifies the maximum number of queries that each thread should make against the database before exiting. A value of **`-1`** indicates that there should be no maximum number of iterations.

Always Disconnect

This indicates whether each thread should maintain persistent connections to the database or use a separate connection for each query.

SQL ModRate

The SQL ModRate job is intended to measure the update performance for relational databases. It is largely similar to the SQL SearchRate job, but rather than allowing the administrator to specify the full SQL statement, it constructs the statement based on the values of parameters given. The general form of the SQL statement that will be constructed is:

`UPDATE` *{table}* **`SET`** *{updateColumn}* **`=`** *{random}* **`WHERE`** *{queryColumn}* **`=`** *{value}*

where *{table}* is the name of the table to be updated, *{updateColumn}* is the name of the column to be updated, *{random}* is a randomly-generated alphabetic string, *{queryColumn}* is the name of the column to be queried to find the rows to update, and *{value}* is the value or range of values to use when locating the rows to update.

The parameters that may be specified when scheduling an SQL ModRate job are:

JDBC Driver Class

This specifies the fully-qualified Java class name that provides the JDBC driver to use to communicate with the database.

JDBC URL

This specifies the JDBC URL to use to connect to the database, in the form "**jdbc:***{dbtype}***:***{driverdata}*".

User Name

This specifies the username of the account to use when connecting to the database.

User Password

This specifies the password to use when connecting to the database.

Table Name

This specifies the name of the database table in which the updates are to be performed.

Column to Update

This specifies the name of the column whose value will be replaced with a randomly-generated string of alphabetic characters.

Generated Value Length

This specifies the number of characters to use in the randomly-generated strings that will be used to replace the values in the column to update.

Column to Match

This specifies the name of the column to use to find the rows to update.

Match Value

This specifies the value to use to find the rows to update. It may be a single value, but it may also contain a bracketed numeric range to vary the value used for each update. For example, a value like

```
user.[1-1000000]
```

will cause the client to choose a value at random between 1 and 1000000 to use for each update. Similarly, replacing the dash with a colon, like

```
user.[1:1000000]
```

will cause the client to choose the values sequentially rather than at random.

Warm Up Time

This specifies the length of time in seconds that the job should be allowed to run before beginning to collect statistics. This makes it possible to ensure that all connections are established and that the load against the database is stable before beginning to measure performance.

Cool Down Time

This specifies the length of time in seconds that the job should be allowed to run after ending statistics collection. This makes it possible to ensure that the load against the database is still stable before ending performance measurement.

Number of Iterations

This specifies the number of updates that each client thread should perform before it exits. A value of **`-1`** indicates that there will be no maximum number of updates.

Always Disconnect

This indicates whether each thread should disconnect from the database after each update or maintain a persistent connection for all operations.

Identity Synchronization for Windows Jobs for Active Directory

SLAMD is provided with a number of jobs intended for testing the Sun Identity Synchronization for Windows software. Many of these jobs are provided for the purpose of interacting with Active Directory. All these jobs communicate with Active Directory using LDAP and therefore they can be executed from any client system. Because clients can communicate with Active Directory using LDAP, many of the standard LDAP job classes provided with SLAMD may be used to interact with Active Directory in some aspects. However, the jobs listed in this section provide special support for Active Directory schema requirements, including creating accounts based on the **`user`** objectClass and using the properly-encoded **`unicodePWD`** attribute for the password which are specific to Active Directory and not standard in other LDAP directories.

The AD ModRate Job

The AD ModRate job provides a means to perform repeated modify operations in an Active Directory server. It provides special support for the encoding required for the **`unicodePWD`** attribute used to hold passwords. The configuration parameters for the AD ModRate job include:

Active Directory Host

This field specifies the address of the Active Directory server. This may be either an IP address or resolvable hostname.

Active Directory SSL Port

This field specifies the port number on which the Active Directory server listens for SSL-based connections. It is necessary to communicate using SSL because

Active Directory will only support password changes if the request is sent over a secure connection.

Bind DN

This field specifies the DN to use to bind to the Active Directory server. The user specified by this DN must have permission to perform the modify operations.

Bind Password

This field specifies the password to use when binding to the Active Directory server.

Entry DN

This field specifies the DN of the entry to modify. This DN can contain a bracketed numeric range to specify that multiple entries should be targeted. For example, the DN

```
cn=User [1-1000],cn=Users,dc=example,dc=com
```

indicates that a number should be chosen at random between 1 and 1000 (inclusive) and used in place of the bracketed range. Alternately, a colon may be used in place of a dash, like:

```
cn=User [1:1000],cn=Users,dc=example,dc=com
```

to indicate that the entries should be modified sequentially rather than chosen at random.

Attribute(s) to Modify

This text area specifies the names of the attributes that should be targeted by the modify operations. If multiple attributes are to be targeted, then the names of those attributes should be provided on separate lines of the text area. The values of those attributes will be replaced with a string of randomly-chosen characters. If multiple attributes are specified, then each will be given the same value. If **`unicodePWD`** is included as one of the attributes to modify, then its value will be properly encoded.

Value Length

This field specifies the number of characters that should be used in the randomly-generated values for the modifications.

Warm Up Time

This field specifies the length of time in seconds that the job should run before beginning to collect statistics. Using a small warm-up time makes it possible to ensure that all client threads have started and the load against the server is stable before beginning to collect statistics.

Cool Down Time

This field specifies the length of time in seconds before the job ends that it should stop collecting statistics. Note that this may only be used if the job ends because the stop time or maximum duration has been reached.

Modify Time Limit

This field specifies the maximum length of time in seconds that will be allowed for a modify operation to complete. If any modify operation takes longer than this period of time, then it will be abandoned and considered failed.

Time Between Requests

This field specifies the length of time in milliseconds that should be allowed between modify requests submitted by a client thread. Note that this time is measured between the beginning of one modify request and the beginning of the next, which makes it possible to target a specific modification rate. If a modify request takes longer than this period of time to complete, then there will not be any delay before the next modify request.

Number of Iterations

This field specifies the total number of modify operations that should be performed by each client thread. A value of -1 indicates that there should not be any limit to the number of modify operations performed.

Always Disconnect

This checkbox is used to indicate whether the client should disconnect from the Active Directory server after each modify operation and establish a new connection for the next update. By default, each thread will establish a connection at the beginning of the job and will use that connection for all modify operations performed by that thread.

The AD AddRate Job

The AD AddRate job is similar to the AD ModRate job with the exception that it performs add operations rather than modifies. The entries added will look like:

```
dn: cn=User 1,cn=Users,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: User 1
givenName: User
sn: 1
samAccountName: user.1
userAccountControl: 544
unicodePWD:: IgBwAGEAcwBzAHcAbwByAGQAIgA=
```

Note that because this job creates entries with sequentially-incrementing numbers, it may only be executed using a single client to ensure that the sequential order is preserved. This should not be a significant limitation because the work it performs does not consume significant client resources, and it is possible to use multiple threads on that client, as the order will be preserved across those clients.

Many of the configurable parameters for the AD AddRate job are the same as for the AD ModRate job. The differences are:

Base DN

This field specifies the DN of the entry below which the user entries will be created. In the example given above, this base DN would be "**cn=Users,dc=example,dc=com**".

Initial User Number

This field specifies the number that should be used for the first user entry created. The user entries created will use numbers incrementing sequentially starting at this value.

Final User Number

This field specifies the number that should be used for the last user entry created. The job will complete once the user entry with this number has been added.

User Password

This field specifies the password that should be used for the user entries that are added to Active Directory.

The AD DelRate Job

The AD DelRate job is intended to remove the entries created by the AD AddRate job. The standard LDAP DelRate job can also be used to accomplish this task, but the AD DelRate job is written to work in conjunction with the AD AddRate job. As such, the parameters for the AD DelRate job are the same as for the AD AddRate job.

The AD Add and Del Rate Job

The AD Add and Del Rate job combines the work of the AD AddRate and AD DelRate jobs into a single job. This job will first add a number of entries and will subsequently delete them. This is provided as a convent alternative to scheduling separate AD AddRate and AD DelRate jobs, and it is also suitable for use as an optimizing job.

Most of the parameters for the AD Add and Del Rate job are the same as for the AD AddRate job. There is, however, one additional parameter:

Time Between Adds and Deletes

This field specifies the length of time in seconds that the client should wait after all adds have been completed before beginning to perform the delete operations.

Identity Synchronization for Windows Jobs for Windows NT

SLAMD also provides a number of jobs for performing operations in a Windows NT domain for the purpose of testing Identity Synchronization for Windows. Unlike later versions of Windows that use Active Directory, Windows NT does not provide the ability for clients to communicate with it using LDAP. It does, however, offer commands that may be executed on the system to create, modify, and remove user accounts, and the ISW jobs that communicate with Windows NT systems use these commands. As such, the jobs for Windows NT must be run on a single client running on the Windows NT server system itself. However that client may use multiple threads to perform the operations.

The NT ModRate Job

The NT ModRate job provides the ability to modify user accounts in the Windows NT domain. In particular, it can modify the comment and/or password with a value containing a timestamp (number of milliseconds since January 1, 1970). It does this using a command like:

```
NET USER userid timestamp /COMMENT:timestamp
```

The configurable parameters for the NT ModRate job class are:

User to Modify

This field specifies the user ID of the user account to modify. It may be a single user ID, but it may also contain a bracketed numeric range to specify multiple users. For example, a user ID of:

```
user.[1-1000]
```

indicates that a random number should be chosen between 1 and 1000 and used in place of the bracketed range. Similarly, using a colon instead of a dash, like

```
user.[1:1000]
```

indicates that the values should be chosen in sequential order rather than at random.

Attribute(s) to Modify

This list box indicates which attribute should be modified. Windows NT offers a very limited number of user attributes, and as such only the comment, user password, or both comment and password may be modified. If both the password and comment are modified, then they will be given the same value.

Time Between Modifies

This field specifies the length of time in milliseconds that should be allowed between modifications performed by each thread. This time is measured between the beginning of one modify operation and the beginning of the next, which makes it possible to target a specific modification rate. If any modify operation takes longer than this length of time to complete, then there will not be any delay before beginning the next modify operation.

Warm Up Time

This field specifies the length of time in seconds that the job should be allowed to run before beginning to collect statistics. This makes it possible to ensure that all client threads are active before beginning statistics collection.

Cool Down Time

This field specifies the length of time in seconds before ending that the job should stop collecting statistics. Note that this may only be used if the job is configured to end because of a stop time or maximum duration.

Number of Iterations

This field specifies the maximum number of modify operations that each client thread should perform. A value of -1 indicates that there should not be any limit to the number of modify operations performed.

The NT AddRate Job

The NT AddRate job may be used to add new user accounts to the Windows NT domain. It does this by executing a command like:

```
NET USER user.1 password /ADD /FULLNAME:"User 1"
```

The NT AddRate job offers the following configuration parameters:

Initial User Number

This field specifies the number that should be used for the first user account created. The user accounts will be added with sequentially-incrementing numbers starting with this number.

Final User Number

This field specifies the number that should be used for the last user account created. Once the account has been created with this value, the job will be complete.

User Password

This field specifies the password that should be used for the user accounts created. All user accounts will be created with the same password.

Time Between Adds

This field specifies the length of time in milliseconds that should be allowed between add requests by each client thread. This time will be measured from the beginning of one add to the beginning of the next, which makes it possible to target a specific number of add operations per second.

The NT DelRate Job

The NT DelRate job can be used to delete user accounts from the Windows NT domain. It does this using a command like:

```
NET USER user.1 /DELETE
```

This job is intended to delete the user accounts created with the NT AddRate job and therefore shares the same configuration parameters as that job.

The NT Add and Del Rate Job

The NT Add and Del Rate job combines the work of the NT AddRate and NT DelRate jobs, first adding a number of entries and then deleting them. This makes it possible to perform these tasks in an optimizing job. The only new configuration parameter used in this job is:

Time Between Adds and Deletes

This field specifies the length of time in seconds that the client should wait after completing all add operations before beginning to delete the entries that were added.

Identity Synchronization for Windows Latency Checking Jobs

The ability to perform modify, add, and delete operations against Active Directory and Windows NT domains can be very useful for a number of reasons, particularly in an environment configured to use Identity Synchronization for Windows. However, the ability to make changes in the endpoints of an ISW environment would be even more useful if there were some way of ensuring that those changes are propagated to other systems. SLAMD provides a number of jobs that offer the ability to make changes in one system and attempt to measure the time required for those changes to appear in another.

The AD to DS Latency Checking Jobs

The AD to DS ModRate with Latency Checking job is used to perform modifications against an entries in Active Directory server and measure the average length of time for those changes to propagate to a Sun ONE Directory Server. It does this by operating in a manner very similar to the AD ModRate job but additionally performing periodic modifications to a separate user entry in the Active Directory server and using a persistent search to detect the application of that change in the Sun ONE Directory.

The AD to DS AddRate with Latency Checking job is similar but performs add operations in the Active Directory server rather than modify operations. However, the latency checking process works in exactly the same way in that it periodically modifies a given entry in the Active Directory server and detects that modification using a persistent search in the Sun ONE Directory. Because the latency checking process is exactly the same as for the AD to DS ModRate with Latency Checking job, all the pertinent configuration parameters are also identical.

The configuration parameters used to customize the latency checking process include:

Active Directory Host

This field specifies the address of the Active Directory server in which to perform the modify operations.

Active Directory SSL Port

This field specifies the port number on which the Active Directory server listens for SSL-based connections. Communication with Active Directory must use SSL because Active Directory requires that password changes using LDAP be made over a secure connection.

Active Directory Bind DN

This field specifies the DN of the user that should be used to bind to the Active Directory server. This user must have permission to perform the modify operations in the Active Directory server, including to modify the latency check entry DN.

Active Directory Bind Password

This field specifies the password that should be used to bind to the Active Directory server.

Sun ONE Directory Host

This field specifies the address of the Sun ONE Directory that is being synchronized with the Active Directory server in which the modifications will be made.

Sun ONE Directory Port

This field specifies the port number that should be used to connect to the Sun ONE Directory. If SSL is to be used to communicate with the Sun ONE Directory, then this should be the secure port of the server.

Use SSL for the Sun ONE Directory

This checkbox indicates whether to use SSL to communicate with the Sun ONE Directory.

Sun ONE Directory Bind DN

This field specifies the DN of the user to use when binding to the Sun ONE Directory.

Sun ONE Directory Bind Password

This field specifies the password to use to bind to the Sun ONE Directory.

Active Directory Monitor Entry DN

This field specifies the DN of the entry that should be periodically modified in the Active Directory server when measuring propagation latency. This user should not be one that is modified during the normal processing of the ModRate job or by anything other than the latency checking process while the job is running.

Sun ONE Directory Monitor Entry DN

This field specifies the DN of the entry in the Sun ONE Directory that is synchronized with the Active Directory monitor entry. This entry should not be modified by anything other than Identity Synchronization for Windows while the job is running.

Latency Check Attribute

This field specifies the attribute that should be modified in the Active Directory server by the latency checking process. This attribute must be one that is synchronized with an attribute in the Sun ONE Directory.

Latency Check Delay

This field specifies the delay in milliseconds between latency checks. This delay is measured from the beginning of one modification of the Active Directory monitor entry to the beginning of the next modification, which makes it possible to target a specific latency check rate. If it takes longer than this length of time to perform the modification and detect that it has propagated to the Sun ONE

Directory, then there will be no delay before starting the next latency check modification.

The DS to AD Latency Checking Jobs

The DS to AD ModRate with Latency Checking job is virtually identical to the AD to DS ModRate with Latency Checking job with the exception that the modify operations are performed against the Sun ONE Directory rather than Active Directory, and therefore the latency checking measures the propagation time in that direction as well. However, since Active Directory does not support the persistent search operation to detect changes to entries, it is necessary to periodically poll the Active Directory server to determine whether the AD monitor entry has been updated. In particular, the value of the **usnChanged** attribute in the Active Directory server is monitored because each modification to an entry increases the value of the **usnChanged** attribute in that entry.

The DS to AD AddRate with Latency Checking job is similar but performs add operations in the Sun ONE Directory rather than modifies. However, the latency checking process is identical.

The DS to AD Template-Based AddRate with Latency Checking job is also similar, but uses a template to define the entry to add to the Sun ONE Directory rather than using the default entry format. This template uses exactly the same syntax and logic as the standard SLAMD Template-Based AddRate job, and that template format is documented in the SLAMD Job Reference Guide. Again, the latency checking process is identical for this job as it is for the DS to AD ModRate with Latency Checking job.

The DS to AD latency checking jobs offer one additional configuration parameter not available in the AD to DS latency checking jobs:

Active Directory Poll Delay

This field specifies the delay in milliseconds between polls of the entry in the Active Directory server that is being synchronized with the Sun ONE Directory monitor entry. This delay is measured from the beginning of one poll operation to the beginning of the next.

The DS to NT Latency Checking Jobs

The DS to NT ModRate with Latency Checking job is one that performs modify operations in the Sun ONE Directory Server and measures the time required for those changes to propagate to a Windows NT domain. It is very similar to the DS to AD ModRate with Latency Checking job, including the fact that it must periodically poll the Windows NT system to determine whether an update has occurred. However, Windows NT does not offer a **usnChanged** attribute like Active Directory, so an alternate polling mechanism must be used. In this case, the value of the comment attribute is monitored. The DS to NT AddRate with Latency Checking job and the DS to NT Template-Based AddRate with Latency Checking job use the same latency checking algorithm as the DS to NT ModRate with Latency Checking job.

Note that because these jobs interact with a Windows NT system and must execute commands on that NT system to detect updates, the jobs may only be executed on a client running on that Windows NT system.

There are a couple of minor differences between the jobs measuring latency between the Sun ONE Directory and Windows NT systems and those that measure latency between the Sun ONE Directory and Active Directory systems. The configuration parameters specific to systems using Windows NT include:

Windows NT Monitor User

This field specifies the user ID of the user that will be periodically polled in the Windows NT system to determine if the corresponding entry in the Sun ONE Directory has been updated.

NT Domain Poll Delay

This field specifies the length of time in milliseconds that should be allowed between polls of the Windows NT monitor user to determine whether that entry has been updated.

DS Attr Mapping to NT Comment

This field specifies the name of the attribute in the Sun ONE Directory that is synchronized with the comment attribute in the Windows NT domain.

The NT to DS Latency Checking Jobs

The NT to DS Latency Checking jobs are similar to the DS to NT latency checking jobs with the exception that the changes occur in the opposite direction. Consequently, it is necessary to detect those changes in the Sun ONE Directory rather than in the Windows NT domain. Because the Sun ONE Directory supports the persistent search control, it is possible to use that mechanism to detect changes to the monitor entry rather than through periodic polling.

No new configuration parameters are added for these jobs. All parameters used to configure latency checking are used in the same way as for jobs measuring latency in other aspects of the ISW environment.

SLAMD License

The SLAMD Distributed Load Generation Engine (SLAMD) is licensed under the Sun Public License Version 1.0. This is an OSI-Approved Open Source license, and more information can be found about such licenses at http://www.opensource.org/, but the full text of the Sun Public License Version 1.0 is provided below for reference.

SUN PUBLIC LICENSE Version 1.0

1. Definitions.

1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof and corresponding documentation released with the source code.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License" means this document.

1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing

Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated documentation, interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1 The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the

origin or ownership of the Covered Code.

3.4. Intellectual Property Matters.

(a) Third Party Claims.

If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL'' which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.

If Contributor's Modifications include an application programming interface ("API") and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.

Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.

You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.

You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the

Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.

You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions.

Sun Microsystems, Inc. ("Sun") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.

Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Sun. No one other than Sun has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.

If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must: (a) rename Your license so that the phrases "Sun," "Sun Public License," or "SPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Sun Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS'' BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR

ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

Exhibit A -Sun Public License Notice.

The contents of this file are subject to the Sun Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License. A copy of the License is available at http://www.sun.com/

The Original Code is _____. The Initial Developer of the Original Code is _____. Portions created by _____ are Copyright

(C)_____. All Rights Reserved.

Contributor(s): _____.

Alternatively, the contents of this file may be used under the terms of the _____ license (the "[___] License"), in which case the provisions of [_____] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [____] License and not to allow others to use your version of this file under the SPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [___] License. If you do not delete the provisions above, a recipient may use your version of this file under either the SPL or the [___] License."

[NOTE: The text of this Exhibit A may differ slightly from the text of the notices in the Source Code files of the Original Code. You should use the text of this Exhibit A rather than the text found in the Original Code Source Code for Your Modifications.]