

# **CA Secure Proxy Server**

Technical Note  
For  
Tuning Parameters inside SPS

## Table of Contents

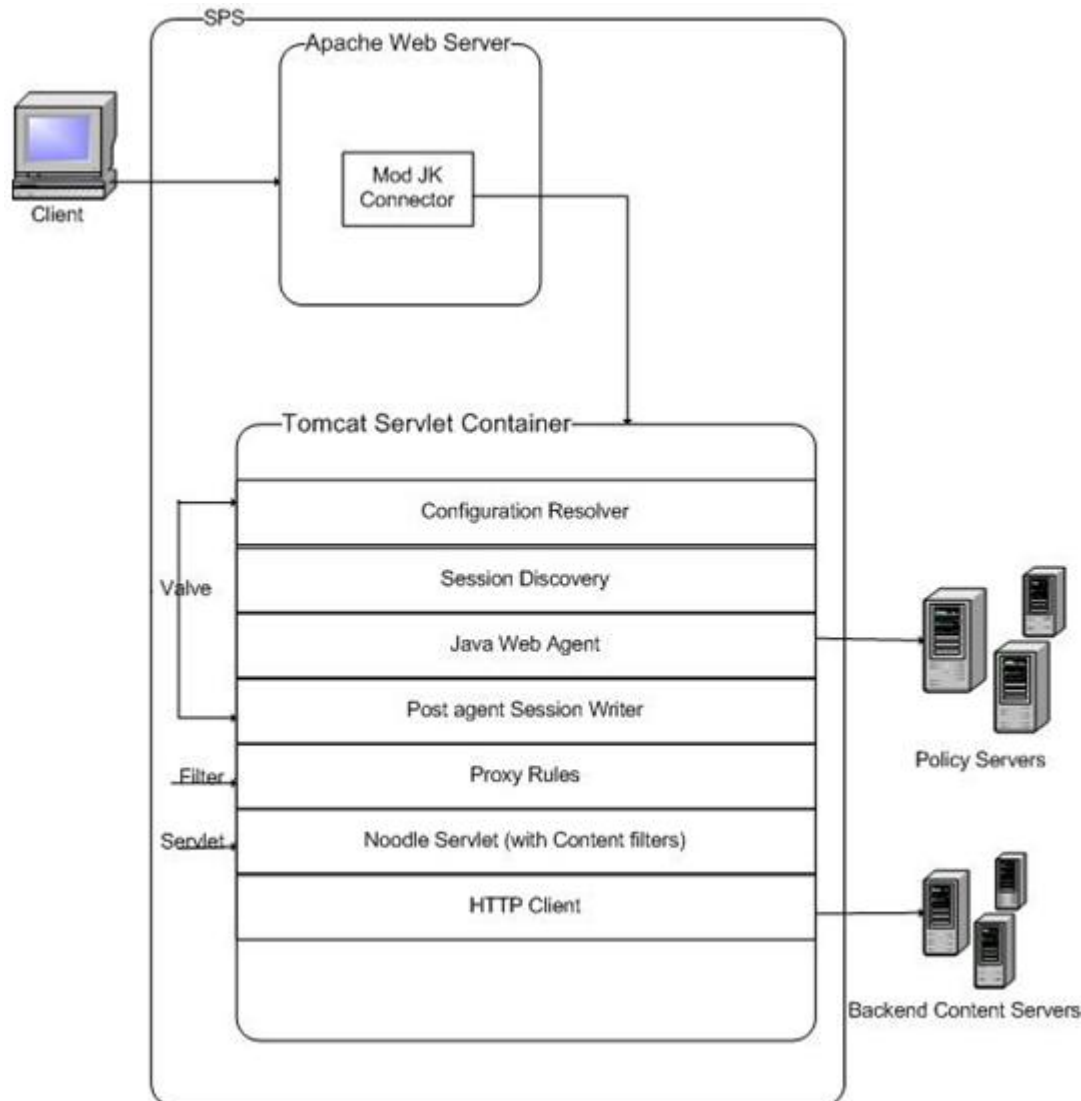
Introduction.....	3
SPS Component Architecture .....	3
Tunable Sections inside SPS.....	4
1. Apache .....	4
2. Tomcat Connector.....	5
3. HttpClient ConnectionPool.....	6
4. JVM.....	8
Tuning Secure Proxy Server for Concurrent Usage .....	9

## Introduction

This technical note describes the tuning parameters that are available inside SPS for tuning SPS performance.

## SPS Component Architecture

SPS architecture consists of an HTTP listener (Apache) and a Tomcat Servlet container. The following illustration shows the major components inside SPS:



The tunable sections of the SPS include Apache, Apache-Tomcat connector, HttpClient ConnectionPool and the JVM.

Apache acts as an Http-Listener inside SPS which forwards the request to embedded Tomcat engine via the mod-jk. The HttpClient ConnectionPool is a pool of connections that SPS maintains for establishing connections with the backend web servers.

## Tunable Sections inside SPS

### 1. Apache

Since Apache (version 2.2.6) is acting as an Http-Listener inside SPS, it can be tuned from the perspective of number of threads or max clients it can support concurrently.

The Apache server ships with a selection of Multi-Processing Modules (MPMs) which are responsible for binding to network ports on the machine, accepting requests, and dispatching children to handle the requests. The MPM modules with which Apache (used inside SPS) has been built are as follows:

For Windows platform: mpm\_winnt (The tunable parameters are available inside httpd.conf)

For UNIX platform: worker (The tunable parameters are available inside httpd-mpm.conf)

Below is the list of the important parameters that need to be tuned from the perspective of number of Threads or max clients it can support:

Parameter	Description	Default Value
ThreadsPerChild	This parameter sets the number of threads created by each child process. The child creates these threads at startup and never creates more. This parameter holds significance for Windows platform since there is only one child process. This parameter should be high enough to handle the entire load of the server. For UNIX platform, where there are multiple child processes, the corresponding parameter would be MaxClients.  Consider increasing this value on Windows if more number of concurrent connections/requests are expected.	250(for Windows)  25(for UNIX)
ThreadLimit	This parameter sets the maximum configured value for ThreadsPerChild for the lifetime of the Apache process.  This value is not present by default in the conf files.	1920(for Windows)  64(for UNIX)
MaxClients	This parameter sets the limit on the number of simultaneous requests that will be served. Once a child process is freed at the end of a different request, the connection will then be serviced. It is only available for UNIX platform  Consider increasing this value if more number of concurrent connections/requests are expected.	150(for UNIX)

Please refer to the <http://httpd.apache.org/docs/2.0/mpm.html> for more details on the configurable parameters that are available with MPM module



## 2. Tomcat Connector

Tomcat ajp13 connector receives the requests that are forwarded by Apache via mod\_jk. The Tomcat initialization is customized so it does not allow deployment of any external applications or servlets. The standard Tomcat xml (server.xml) is not used for initialization. The tunable parameters for Tomcat Ajp13 Connector are available inside server.conf (<sps\_home>/secure-proxy/proxy-engine/conf/server.conf). Below is the list of the parameters:

Parameter	Description	Default Value
worker.ajp13.accept_count	Number of request waiting in queue (queue length): This represents the maximum queue length for incoming connection requests when all possible request processing threads are in use. Any requests received when the queue is full are refused.	10
worker.ajp13.min_spare_threads	Number of threads created at initialization time: This represents the number of request processing threads that will be created when this connector is initialized.  This attribute should be set to a value smaller than that set for worker.ajp13.max_threads	10
worker.ajp13.max_threads	Maximum number of concurrent connections possible: This represents the maximum number of request processing threads to be created by this connector, which therefore determines the maximum number of simultaneous requests that can be handled.	100
worker.ajp13.reply_timeout	The maximum time (milliseconds) that can elapse between any two packets received from proxy engine after which the connection between HTTP listener and proxy engine is dropped. A value of zero makes it to wait indefinitely until response is received (default)  The parameter value should be kept equivalent to the http_connection_timeout.	0 (infinite/ never timeout)
worker.ajp13.retries	The maximum number of times that the worker will send a request to Proxy Engine in case of a communication error. Each retry will be done over another connection. The first time already gets counted, so retries=2 means one retry after error	2

### 3. HttpClient ConnectionPool

The HTTP client component is used by the SPS to send requests over the wire to the backend server and receive responses from the backend server. HttpClient ConnectionPool is a pool that maintains the connections that are made with the backend web server by HttpClient component. Below is the list of the tunable parameters that are available inside server.conf (<sps\_home>/secure-proxy/proxy-engine/conf/server.conf).

Parameter	Description (Note that below settings are per backend webserver)	Default Value
http_connection_pool_min_size	The minimum number of connections to a single destination server that will be maintained by the SPS and available for processing user requests. It defines the minimum number of connections that would be created when the first request is made to a particular backend web server.	4
http_connection_pool_max_size	The maximum number of connections between the SPS and a destination server. It defines the maximum number of connections that can be made with a particular backend web server at a given point of time. These connections would be available in the pool. If all the connections are busy then the request will have to wait for any of the connections to be available.  Note: Each connection established by the SPS creates a socket. For UNIX operating systems, if the maximum size of the connection pool is large, you might need to increase the operating system's limit on file descriptors to accommodate the large number of sockets	20
http_connection_pool_incremental_factor	The increment factor when creating the connections. It defines the number of connections that would be added (in case the http_connection_pool_max_size has not reached) to the pool if all the connections that are already present in the pool are being used to process requests.	4
http_connection_pool_connection_timeout	This parameter defines the Idle time for a connection (in minutes or seconds as specified by the parameter http_connection_pool_connection_timeout_unit) after which the connection is closed and removed from the pool.	1
http_connection_pool_connection_timeout_unit	This parameter defines whether the value specified in the http_connection_pool_connection_timeout is to be considered in minutes or seconds.	minutes

	Valid values are “minutes” or “seconds”	
http_connection_pool_wait_timeout	<p>Timeout (in milliseconds) to be used to wait for an available connection.</p> <p>If all the connections present in the pool are busy at a given point of time and the http_connection_pool_max_size limit has reached then the request would wait for this much amount of time before getting timed out.</p> <p>A timeout of zero means wait indefinitely.</p>	0 (infinite/ never timeout)
http_connection_pool_max_attempts	<p>Number of attempts to obtain a connection from the pool. A value of zero causes it to attempt indefinitely.</p> <p>This parameter is only applicable if wait timeout is not zero</p>	3
http_connection_timeout	<p>Timeout (in milliseconds) to be used for creating connections and reading responses.</p> <p>This parameter defines the socket timeout as well as it limits the time that would be spent in establishing a connection to backend web server. This would include the time spent doing the host name translation and establishing the connection with the backend web server when creating sockets.</p> <p>A timeout of zero means wait indefinitely.</p> <p>The parameter value should be kept equivalent to the worker.ajp13.reply_timeout.</p>	0

## 4. JVM

The JVM tuning parameters can have major impact on performance especially on UNIX. The JVM settings can be tuned via the `SmSPSProxyEngine.properties` (`<sps_home>/secure-proxy/proxy-engine/conf/SmSPSProxyEngine.properties`) on Windows platform and via the `proxyserver.sh` (`<sps_home>/secure-proxy/proxy-engine/proxyserver.sh`) for UNIX platform.

Most important parameters are `-Xms` (default `--256m`) and `-Xmx` (default `--512m`) to define proper memory usage by SPS

**-Xms** - This setting tells the JVM to set its initial heap size. By telling the JVM how much memory it should initially allocate for the heap, we save it from growing the heap size frequently

**-Xmx** - This setting tells the JVM, the maximum amount of memory it should use for the heap. Placing a hard upper limit on this number means that the Java process cannot consume more memory than physical RAM available. This limit can be raised on systems with more memory.

Note: Do not set this value to near or greater than the amount of physical RAM in your system or it will cause severe swapping during runtime.



## Tuning Secure Proxy Server for Concurrent Usage

Proper Tuning of Secure Proxy Server for concurrent usage is recommended so as to get the optimum performance. The important factors that need to be considered for this are the user load expected on the SPS, the speed of the backend web servers or the response time of the applications deployed at the backend web servers.

The number of simultaneous requests that will occur in a given system is most largely dependent on the speed (response time) of the backend applications. Below is an example

- 100 virtual users are requesting a series of small 100 byte pages as fast as they can, will not cause the SPS to have to handle more than 5-10 concurrent requests.
- The same 100 virtual users requesting a CGI that takes 1-2 seconds to respond will create close to 100 concurrent requests.

The above example shows that if backend applications are slow then many simultaneous requests can occur even at times of relatively light traffic. The parameters that determine the maximum number of concurrent request that SPS can handle would be `worker.ajp13.max_threads` and the `http_connection_pool_max_size` (per backend web server). Also Apache needs to be configured for handling this many concurrent requests.

Since the request made to the SPS is intercepted by Apache first it should be the limiter for the number of concurrent requests that SPS can handle. If the concurrent requests coming to SPS are more than the max requests that Apache can handle then it can give a friendly error to the end user and not forward the request to Tomcat

The timeout parameters (Tomcat/ajp13 timeout parameter and `HttpConnectionPool` timeout parameters) available inside `server.conf` also play an important role. These timeout values have been left at 0 (never timeout) by default because their values will entirely be dependent on the response time of the applications that are being hosted at the backend webserver.

If one of the backend servers for which SPS has been configured to forward the request is slow or unresponsive it could affect the performance of SPS if timeouts are not configured properly, i.e left as default. This slow backend server will not only eat up the `HttpConnectionPool` connections but also start eating up the Tomcat/ajp13 connections since the previous requests would be unresponsive for a large amount of time and the new requests for the same backend web server will start getting queued up waiting for the available connection from the `HttpConnectionPool`. This could finally lead to SPS becoming unresponsive since all the Tomcat/Apache connection would get eaten up. The timeout parameters available in `server.conf` will help in such a situation.

If connections in the `HttpConnectionPool` reach to the max limit, the newer requests will start to queue up waiting for the connection to be available. These newer requests will, however, continue to consume the Apache-Tomcat connections that could in turn lead to exceeding the limit set for Apache-Tomcat connections. The `http_connection_pool_wait_timeout` parameter, if appropriately used, will help time out these queued up requests, thereby preventing the Apache-Tomcat connections to go beyond the limit.



The `http_connection_timeout` parameter helps to define the max time that would be spent in establishing a connection to backend web server. If appropriately used, this parameter will also time out those connections which have been waiting for long for the response from the backend web server. Setting these timeout parameters judiciously can lead to considerable improvement in performance of Secure Proxy Server. Please refer to the above tables for description of the timeout parameters

Other factors such as a slow policy server installation and slow DNS resolution can also affect the performance of Secure Proxy Server. These are more of an enterprise issues than something you can control on the SPS.