Article
Mark Bolinsky · Feb 5, 2019 9m read

Apache HTTPD Web Server Configuration for HealthShare

There are often questions surrounding the ideal Apache HTTPD Web Server configuration for HealthShare. The contents of this article will outline the initial recommended web server configuration for any HealthShare product.

As a starting point, Apache HTTPD version 2.4.x (64-bit) is recommended. Earlier versions such as 2.2.x are available, however version 2.2 is not recommended for performance and scalability of HealthShare.

Apache Configuration

Apache API Module without NSD

HealthShare requires the installation option Apache API Module without NSD. The version of the dynamically linked modules depends on the Apache version:

- CSPa24.so (Apache Version 2.4.x)

The configuration of Caché Server Pages in the Apache httpd.conf is best left to be performed by the HealthShare installation that is detailed further on in this document. However the configuration can be performed manually. For more information please see Apache Configuration Guide in the InterSystems documentation: Recommended Option: Apache API Module without NSD (CSPa24.so)

Apache Multi-Processing Module (MPM) Recommendations

Apache Prefork MPM Vs. Worker MPM

Apache HTTPD web server comes with three Multi-Processing Modules (MPM) - Prefork, Worker, and Event. The MPMs are responsible for binding to network ports on the machine, accepting requests, and dispatching children to handle the requests. By default, Apache is usually configured with Prefork MPM which does not scale well for high transaction or high concurrent user workloads.

For HealthShare production systems, the Apache MPM ***Worker*** should be enabled for performance and scalability reasons. Worker MPM is preferred because of the following:

- Prefork MPM uses multiple child processes with one thread each and each process handles one connection at a time. When using Prefork, concurrent requests suffer because as each process can only deal with a single request at a time the requests are queued until a server process becomes free. Additionally in order to scale, more Prefork child processes are required which consumes significant amounts of memory.
- Worker MPM uses multiple child processes with many threads each. Each thread handles one connection at a time, which is big help for concurrency and reduces memory requirements. Worker handles concurrency better than Prefork because there will usually be free threads available to serve the requests instead of single threaded Prefork processes which may be busy.

Apache Worker MPM Parameters

By using threads to serve requests, Worker is able to serve a large number of requests with fewer system resources than the Prefork process-based server.
The most important directives used to control Worker MPM are **ThreadsPerChild** which controls the number of threads deployed by each child process and **MaxRequestWorkers** which controls the maximum total number of threads that may be launched.
The recommended Worker MPM common directive values are detailed in the table below:

Recommended Apache HTTPD Web Server Parameters	Apache Worker MPM Directives	Recommen
	MaxRequestWorkers	Maximum # users, or th sum of all i defined inte * *Note: If al a value of '*
	MaxSpareThreads	250
	MinSpareThreads	75
	ServerLimit	MaxReques

Apache Worker MPM Directives	Recommended Value	Comments
	StartServers	20
	ThreadsPerChild	25

For more information please see the relevant Apache version documentation:

- Apache 2.4 MPM Common Directives

Example Apache 2.4 Worker MPM Configuration

This section details how to configure Worker MPM for a RHEL7 Apache 2.4 web server required to support up to 500 TrakCare concurrent users.

1. First check the MPM using the following command:
```
$ apachectl -V | grep MPM
Server MPM:     prefork
```
2. Edit the configuration file /etc/httpd/conf.modules.d/00-mpm.conf as required, by adding and removing the comment character # so that only the Worker MPM modules is loaded. Modify the Worker MPM section with the following values in the same order as below:
```
# Select the MPM module which should be used by uncommenting exactly one of the following LoadModule lines
# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
#
# worker MPM: Multi-Processing Module implementing a hybrid
# multi-threaded multi-process web server
# See: http://httpd.apache.org/docs/2.4/mod/worker.html
LoadModule mpm_worker_module modules/mod_mpm_worker.so
# Worker MPM paramaters
ServerLimit          20
StartServers         10
MaxRequestWorkers    500
MinSpareThreads      75
MaxSpareThreads      250
ThreadsPerChild      25
```
3. Restart Apache
4. After successfully restarting Apache, validate the worker processes by executing the below commands. You should see something similar to the following confirming the httpd.worker process:
```
$ apachectl -V | grep MPM
Server MPM:     worker
```

Apache Hardening

Apache Required Modules

The installation of the official Apache distributed package will enable a specific set of Apache modules by default. This default configuration of Apache will load these modules into each httpd process. It is recommended to disable all modules that are not required for HealthShare for the following reasons:

- reduce the httpd daemon process footprint.
- reduce the chance of a segmentation fault from a rogue module.
- reduce security vulnerabilities.

The table below details the Apache modules recommended for HealthShare. Any module that is not in the below list can be disabled:

Module Name	Description
alias	Mapping different parts of the host file s redirection.
authzhost	Provides access control based on client
dir	Provides for trailing slash redirects and
headers	To control and modify HTTP request an
logconfig	Logging of the requests made to the ser
mime	Associates the requested file name's ex
negotiation	Provides for content selection of the doc capabilities.
setenvif	Allows for setting of environment variabl
status	Displays server status and performance

Disable Modules

Unnecessary modules should be disabled to harden the configuration that will reduce security vulnerabilities. The customer is responsible for web server security policy. At a minimum the following modules should be disabled.

Modul Name	Description
asis	Sends files that contain their own HTTP
autoindex	Generates directory indices and display present
env	Modifies the environment variable passe
cgi	cgi - Execution of CGI scripts
actions	Executes CGI scripts based on media ty

Modul Name	Description
	requests
include	Server-parsed HTML documents (Serve
filter	Smart filtering of requests
version	Handling version information in config fi
userdir	Mapping of requests to user-specific dir translated to a directory in server

Apache SSL/TLS

To protect data in transit, ensure confidentiality and authentication InterSystems recommends all TCP/IP communication between HealthShare servers and clients be encrypted with SSL/TLS, and InterSystems also recommends HTTPS be used for all communication between the users' browser client and the web server layer of the proposed architecture. Please be sure to consult your organization's security policies to ensure compliance with any specific security requirements of your organization.

The customer is responsible for supplying and managing the SSL/TLS certificates.
If using SSL certificates then add sslmodule (modssl.so).

Additional Hardening Apache Parameters

To further harden the Apache configuration, make the following changes in the httpd.conf:

- TraceEnable should be turned off to prevent potential Cross Site Tracing issues.

```
#Disable HTTP TRACE method
TraceEnable Off
```

- ServerSignature should be turned off so the web server version is not displayed.

```
#Configures the footer on server-generated documents, useful for debug purpose
ServerSignature Off
```

Supplemental Apache Configuration Parameters

Keep-Alive

The Apache Keep-Alive setting allows the same TCP connection for HTTP communication to be used instead of opening a new connection for each new request, i.e. Keep-Alive maintains a persistent connection between client

and server. When Keep-Alive option is enabled performance improvements come from reduced network congestion, reduced latency in subsequent requests, and less CPU usage caused by opening connections simultaneously. Keep-Alive is ON by default and HTTP v1.1 standard mandates that it should be presumed on.

However there are caveats to enabling Keep-Alive; Internet Explorer must be IE10 or higher to avoid known timeout issues with older versions of IE. Also intermediaries like firewalls, load balancers and proxies etc. can interfere with 'persistent TCP connections' and can result in unexpected closure of connections.

When enabling Keep-Alive, the Keep-Alive timeout also needs to be set. The default Keep-Alive timeout for Apache is too low and needs to be increased for most configurations because issues may arise associated with broken AJAX (i.e. hyperevent) requests. These issues can be avoided by ensuring the Keep-Alive timeout on the server is greater than on the client. In other words the client should timeout and close the connection rather than the server. Problems occur – mostly in IE but to a lesser extent in other browsers – when the browser tries to use a connection (particularly for a POST) that it expects to be open.

See below for recommended KeepAlive and KeepAliveTimeout values for a HealthShare web server.

To enable KeepAlive in Apache, make the following changes in the httpd.conf:

```
# KeepAlive: Enables HTTP persistent connections
KeepAlive On


# KeepAliveTimeout: Number of seconds to wait for the next request from the# same client on the same connection.
# KeepAliveTimeout 65 was chosen because the default Internet Explorer KeepAliveTimeout is 60 seconds.
KeepAliveTimeout 65
```

CSP Gateway

For CSP Gateway KeepAlive parameter, leave the value at the default No Action because the KeepAlive status is determined by the HTTP response headers for each request.

#Best Practices #InterSystems Business Solutions and Architectures #Performance #Red Hat Enterprise Linux (RHEL) #SOAP #System Administration #HealthShare