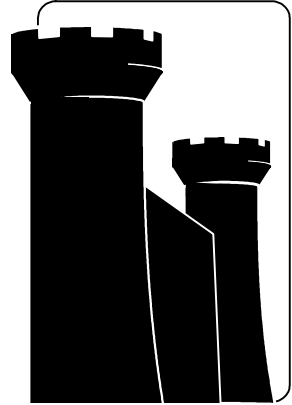

STRONGHOLD WEB SERVER 3.0
Setup Guide



Copyrights

©2000 C2Net Software, Inc.

This manual, as well as the software described in it, is furnished under license and may only be used or copied in accordance with the terms of such license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by C2Net. C2Net assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

Except as permitted by such license, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without prior written permission of C2Net.

Portions of this manual were developed by the Apache Group, and are taken with permission from the *Apache 1.3 User's Guide* at <http://www.apache.org/docs/>.

Portions of this manual were developed by the PHP Documentation Group, and are taken with permission from the PHP3 Manual at <http://www.php.net/manual/>.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>). Some of the cryptographic software in this product was written by Eric Young (ey@cryptsoft.com). Remainder copyright C2Net Software, Inc., 1995, 1996, 1997, 1998, 1999, 2000 all rights reserved.

Stronghold Web Server Administration Guide

Trademarks

Stronghold is a registered trademark of C2Net Software, Inc. C2Net and the C2Net logo are trademarks of C2Net Software, Inc.

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. C2Net cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

C2Net Software, Inc.
1440 Broadway, Suite 700
Oakland, CA 94612 USA
<http://www.c2.net>



Contents

Contents	iii
Installation	5
System Requirements	6
Installing Stronghold Web Server	7
Configuring Stronghold	11
Generating a New key	15
Requesting a Signed Certificate	16
Creating a temporary certificate	18
Manually Upgrading to Stronghold 3.0	20
Upgrading from Stronghold 2.3 or Earlier	20
Upgrading From Stronghold 2.4	23
Upgrading from Apache	25
To upgrade from an Apache Web Server	25
Requesting and Installing New License Blocks	27
To install a license block with install_lb	27
To install a license block manually	27
Default Configuration	29
Glossary	57

Installation

This chapter provides information about how to set up your new Stronghold Web Server, including:

- [System Requirements](#)
- [Installing Stronghold Web Server](#)
- [Manually Upgrading to Stronghold 3.0](#)
- [Requesting and Installing New License Blocks](#)

After installation, you'll probably want to configure the Stronghold Web Server. The Administration Guide provides instructions for using the text-based configuration file and explanations of each individual configuration directive. It also describes how to recompile the server with a different set of modules.

If you are already experienced in configuring and administering Apache web servers then you are probably able to configure Stronghold Web Servers and most of its modules.

System Requirements

Stronghold Web Server is available for most varieties of UNIX operating system. In addition to a version suitable for your operating system, you need a server platform that meets these system requirements:

- at least 90MB available hard disk space.
- at least 16MB available RAM.

These are the minimum requirements for running Stronghold itself. In addition, you also need:

- adequate disk space for HTML, CGI, and log files.

As your log files accumulate data over time, they occupy more disk space. The largest log file is the access log, which occupies about 80 bytes per request, or 1MB per 10,000 requests. For example, if your site receives 200,000 hits per day, you accumulate 20MB of log entries per day. You should plan to rotate your logs regularly to control the file size of the current log, and to allow archiving of older log files to conserve disk space.

- adequate RAM to support your anticipated server load.

If you plan to run a high-traffic site, you need significantly more memory than the minimum. For moderate traffic on a CISC-chip-based platform, we recommend allowing at least 32MB of memory for the server. On a RISC-chip-based platform, allow at least 64MB. If you plan to run a light-traffic site, less memory will suffice.

- platform support for virtual hosts (recommended but not required).

Installing Stronghold Web Server

Stronghold Web Server is distributed in a single, self contained installer file when you download it.

In Stronghold 3.0, as with previous versions no upgrade scripts are available. This version of Stronghold must be installed from scratch.

If you have an existing Apache or Stronghold server, install this version into a new directory and then upgrade manually. See [Manually Upgrading to Stronghold 3.0 on page 10](#).

The installer prompts you for information about your existing system. To guide you through the available options, this section provides a flow chart overview of the installation. The flow chart shows three different elements of the installation procedure:

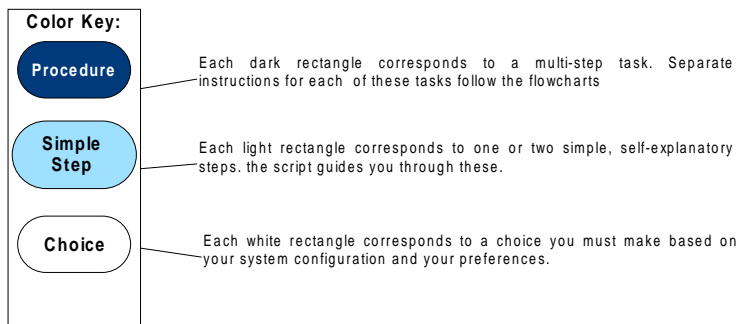


Fig: 1-1 Elements of an Installation Flowchart

The program installs Stronghold Web Server from scratch, then starts the new server with the configuration file shown in Default Configuration at the end of this guide. The minimum configuration required to run the server includes:

- some basic information about your site

The installer uses this information to create a new configuration file.

- an encryption key pair

The installer guides you through the process of generating the key pair.

- an authentication certificate

The installer guides you through the process of requesting a valid certificate, then creates a temporary certificate for the interim. If you already have a certificate and key, or you will be obtaining and configuring them separately, you can bypass the certificate request generation and just let the installer generate a temporary certificate. This will allow the installer to create a functioning installation so it can start up Stronghold. After this you can make the modifications you require to your Stronghold configuration.

The installer prompts you for the required information as it performs its tasks. The dialog between the installer and the user who runs it looks like the following:

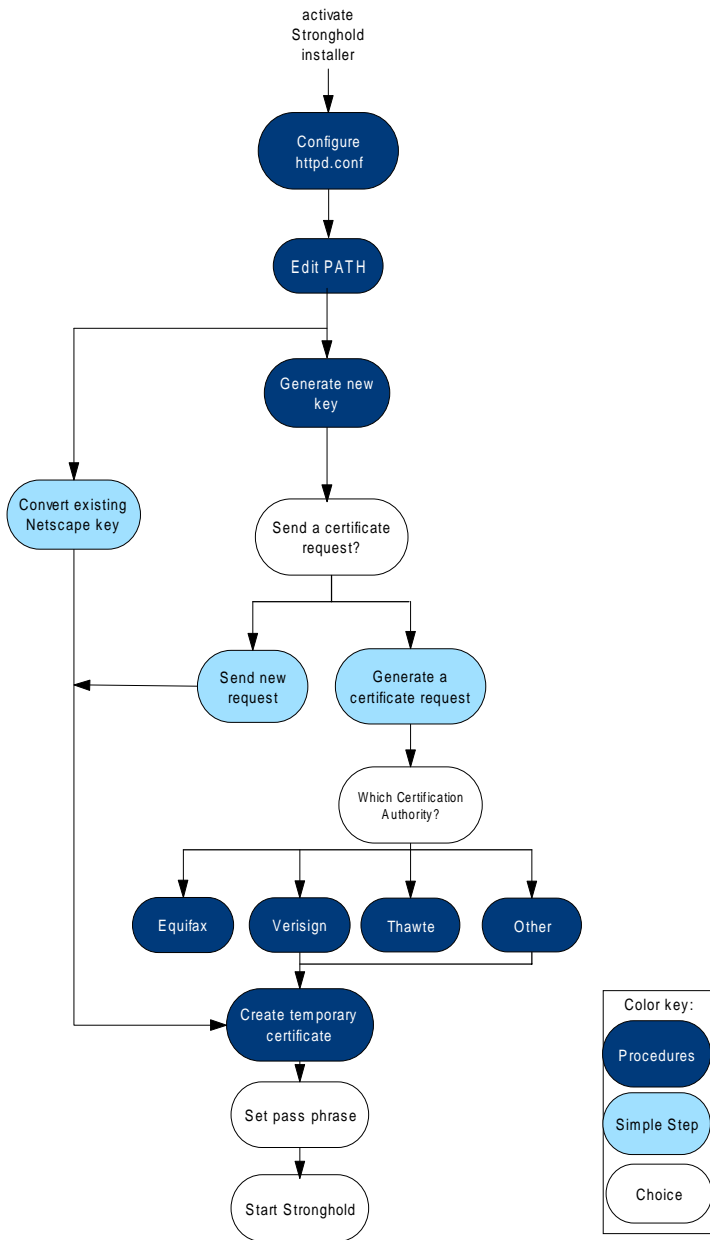


Fig: 1-2 Installing Stronghold Web Server from scratch

To begin the process, move to the directory where the installer program is, then `su` to the user that you want to install the server as. This should be the same user that will start the server, usually root. Start the script from the command line:

```
./sh-30-<platform>
```

if you receive an error, such as “Permission denied,” then you must change the installer’s file permissions before continuing:

```
chmod u+30 sh-30-<platform>
```

If you have an existing server on the same host, you must stop the old server or install Stronghold on ports that your server does not use.

The installer launches with a text-based interface that begins by asking you to confirm that you want to proceed with the installation process:

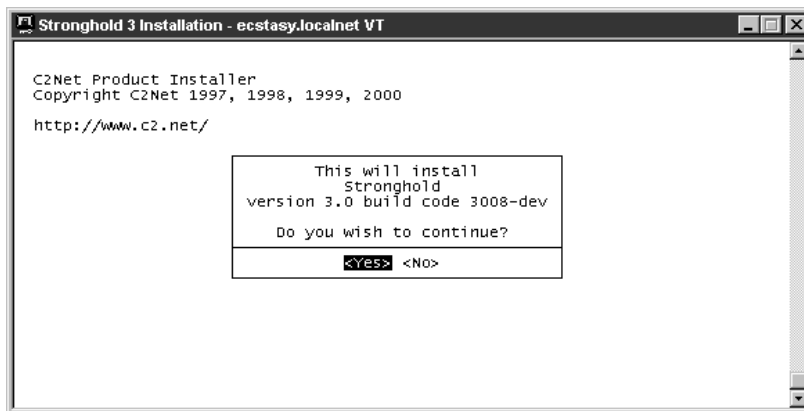


Fig: 1-3 The Stronghold Installer

Select yes by pressing return.

Configuring Stronghold



Fig: 1-4 Absolute path request

1. Enter the [absolute path](#) to the directory where you want to install the server and its components. If the directory you enter does not exist, the program asks you to confirm that you want to create a new directory. In that case, select yes to continue.

This directory is called `<ServerRoot>` . Wherever the documentation refers to `<ServerRoot>`, substitute the actual path to this directory.

The program installs the Stronghold Web Server and its components. A status bar displays the installer's progress in real time. When installation is complete, the program prompts you to press any key to continue.



Fig: 1-5 Logs directory request

2. Enter the path to the directory where you want to store the server logs.

The path can be an [absolute path](#) or a [relative path](#) to the directory you chose for installation. The default is `<ServerRoot>/logs`, but you can use any directory.

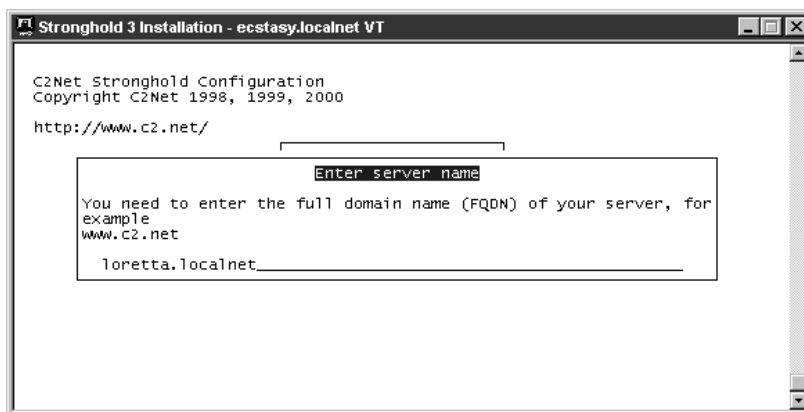


Fig: 1-6 Domain name request

3. Enter the fully qualified Domain name of your main server host.

The installer guesses the hostname and presents this as the default, edit this preselected hostname if it is not correct.

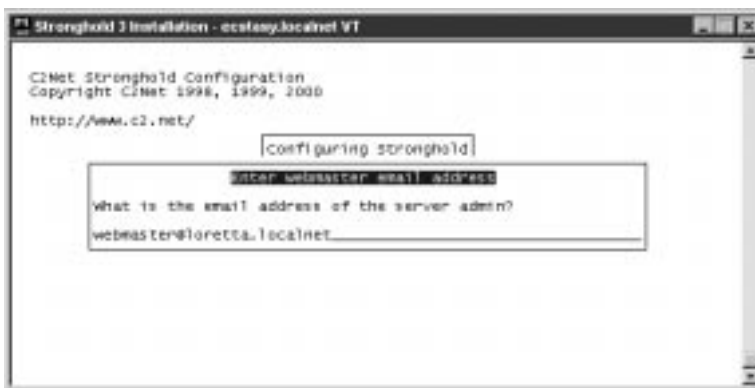


Fig: 1-7 Webmaster email address request

4. Enter the email address of the server administrator.

The default is “Webmaster” at the hostname you entered in the previous step.

5. Enter the number of the port you want to use for regular, unsecured transactions.

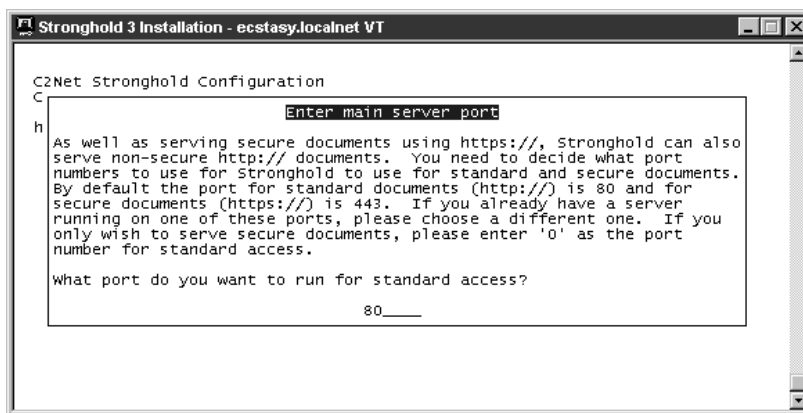


Fig: 1-8 Unsecure port number request

The default is port 80. Browsers automatically direct their requests to port 80 unless the user specifies a different port. To install Stronghold as an SSL-only server, enter “0.”

If the port you choose is already in use, the script prompts you to choose a different one. If you have another Web server running on this platform, select a different port for

Stronghold Web Server or your other server. If you do this, you must advertise this fact to your users

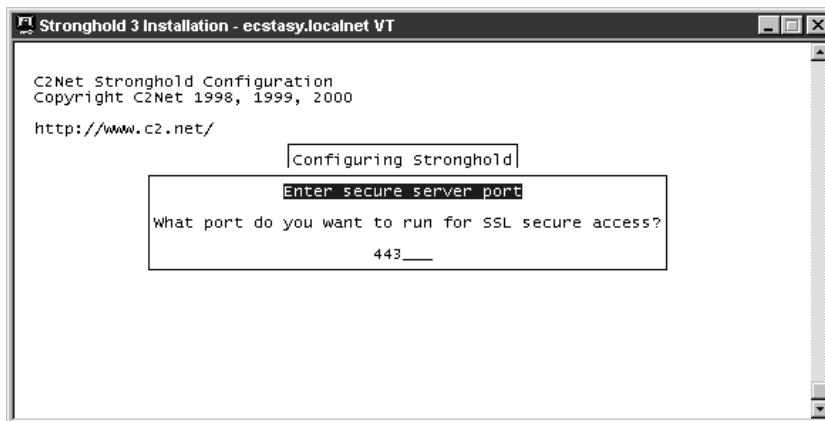


Fig: 1-9 Secure port number request

6. Enter the port you want to use for secure, encrypted transactions.

The default port is 443. Browsers automatically direct HTTPS requests to port 443 unless the user specifies a different port. If you are installing the server as root, you can choose any unused port number. If you are installing as a user other than root, you may be able to only choose an unused port number of 1024 or above.

If the port you choose is already in use, the script prompts you to choose a different one. If you have another Web server running on this platform, select a different port for Stronghold Web Server or your other server. If you do this, you must advertise this fact to your users.

The program warns you that you are about to be prompted for the Stronghold license block that you received via email when you downloaded the installer. If you do not have a license block, contact the organization that sold you Stronghold:

C2Net Software US	C2Net International
stronghold-admin@c2.net	sales@eu.c2.net

7. Paste your complete Stronghold license block into the window.

Your license block looks something like this:

```
*****BEGIN LICENSE BLOCK*****
TEIAAQCKAAAAAGar2a7Pc0iD2oyGDN9a5mrYkZ0NrG7Zcy7UFbFoP4xLzvcVAN4K
0Mrww4z0A2o+gfvADbDz9IdePDKOA21C2E8SiPu1qwVI1pwvbA6xuVRWDo05BT/I
rNGExtSx+LMh3N1q2icj4eD53kNQVoLvaoQ5CcYvWGexrKiSDQvE3agLU2VyaWfS
OiAzMTQxNTkyNjUzNTg5NzkzMjMKQ29tcGFueTogQzJOZXQgRXVybSchmunkZApF
eHBpcmVzOiBOZXZlcgpQcm9kdWN0OiBTSDIKVHlwZfogRXZhHVhdG1vbGpJREVB
OiBZZXMKRmFrZTogCkYyb2t1biBMaW51CkYyb2t1bjoGMQ==
*****END LICENSE BLOCK*****
```

8. Press control-D at the beginning of the first blank line after the license block. Press return to get a blank line if one was not inserted when you pasted the license block.

You may need to wait a moment while the installer processes your license block. When your license block has been accepted, the installer invokes the [genkey](#) utility and exits. [genkey](#) begins by generating a new key pair, as described in the next section.

Generating a New key

1. Enter a key size in bits.

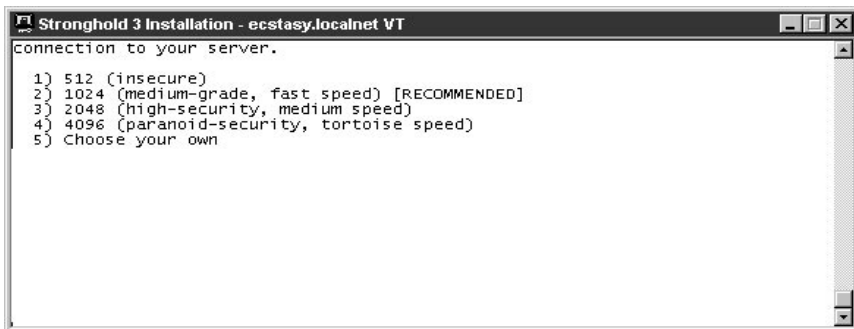


Fig: 1-10 Key size request

Key sizes must either be 512 or 1024 for compatibility with certain browsers. We recommend 1024 bits. Keys larger than 1024 bits are incompatible with some versions of Netscape Navigator and Microsoft Internet Explorer, and with other browsers that use RSA's BSAFE cryptography toolkit.

The program generates some random data with which to create a unique key pair.

2. Enter some random keystrokes when you are prompted to do so.

The program generates more random data based on the intervals between your keystrokes. It also displays a counter that moves closer to zero as you type. Stop when the counter reads 0 and the program beeps:

```
0 * -Enough, thank you.
```

The program generates the key pair and saves it at `<ServerRoot>/ssl/private/<hostname>.key`. Then it asks whether you want to request a signed certificate from a [Certification Authority \(CA\)](#) or not. You must have a signed certificate in order to authenticate your site. to users You can request one from Verisign, Thawte, or another CA.

Requesting a Signed Certificate

1. At the Certification Authority prompt, enter your preferred choice.
2. Enter the two-letter code for your country.

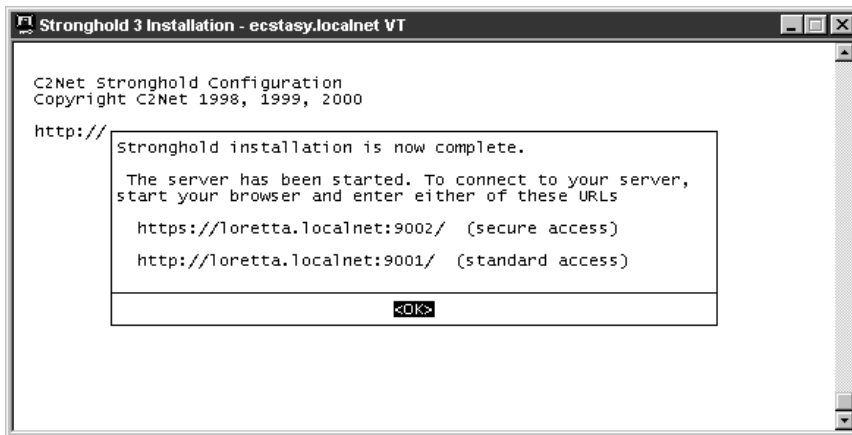


Fig: 1-12 Installation complete.

Creating a temporary certificate

1. At the Certification Authority prompt, enter your preferred choice.
2. Enter the two-letter code for your country.

For a list of valid country codes, see Appendix D of the Administration Guide.

3. Enter the full name of your state or province, the name of your city, town or other locality, the name of your organization, and then enter the name of your unit within the organization.
4. Enter the fully-qualified domain name of your site.

If your preferred CA requires you to supply a challenge password or an optional company name, enter them when prompted otherwise press enter.

The program prints the certificate signing request (CSR), which looks something like this:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBezCBzgIBADB7MQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcml5pYTEQ
MA4GA1UEBxmHT2FrbGFuZDEbMBkGA1UEChMSQzZlZGZGZGZGZGZGZGZGZGZGZGZG
DgYDVQQLEwdUZXN0aW5nMRYwFAYDVQQDEw1nYWJjZGZGZGZGZGZGZGZGZGZGZGZG
hvcNAQEBBQADAwAwOAIxAKJukoQhq4LanG2k+ /LnRTGJAcgv9LJPsdFcsjqRs8ygo
yaw4ucOEdx+WdnM0x36NcQIDAQABMA0GCSqGSIb3DQEBAUAzEABRLR6IkG70oN
G1MnvuMDeWou4kIvc98ysjssCNKsDKsHAXBSEbfsIQs5JRNagVBW
```

-----END NEWCERTIFICATE REQUEST-----

5. Copy and paste this certificate request into the ordering form at your preferred CA, and follow any other specific instructions they may have.

Manually Upgrading to Stronghold 3.0

Upgrades to this version can only be performed manually. The manual upgrade procedure is slightly different depending on whether you are upgrading from Stronghold 2.3 or earlier, Stronghold 2.4, or Apache.

Many third-party modules have not been thoroughly tested for use with Stronghold 3.0 and [Apache](#) 1.3.12. If your existing server has third-party modules that you wish to compile into Stronghold 3.0, do so with caution. When using an existing Configuration file or modifying the new Configuration file to include new modules, note these changes as of Stronghold 2.4 (for those upgrading from Stronghold 2.3 or earlier):

- The source code has been reorganized; modules are now located in subdirectories of a new `<ServerRoot>/src/modules/` directory.
- The `Module` directive has been changed to `AddModule`.
- The `EXTRA_LFLAGS` variable has been changed to `EXTRA_LDFLAGS`.
- The `-DMAXIMUM_DNS` definition is now obsolete.
- The `-DSERVER_SUBVERSION=\"string\"` compile-time option has been replaced with the run-time API call `ap_add_version_component()`.
- The `mod_dir` module has been divided into `mod_dir` and `mod_autoindex`.
- The `mod_auth_mysql` module is no longer included in the distribution.

And the following changes as of Stronghold 3.0:

- The `mod_php` (2.0) module has been replaced by PHP 3.0.16 which is loaded as a DSO.
- `Mod_perl` is precompiled and has been included in the distribution.
- C2Net's `modssl` SSL/TLS module has been removed and replaced with Ralf s. Englishall's `mod_ssl`. Backward compatibility directives have been added allowing Stronghold 2.4.x installations to continue to work as before with no or only minimal changes.

Upgrading from Stronghold 2.3 or Earlier

Upgrading to Stronghold Web Server 3.0 from Stronghold 2.3 or earlier involves:

- copying all customized files from the old server directory tree to the new Stronghold 3.0 directory tree.
- adding custom configurations from old server configuration file to new server configuration file.
- modifying the added custom configurations in the new server configuration file to reflect new configuration schemes introduced in Apache 1.3 and Stronghold 2.4.

-
- configuring Stronghold 3.0 SSL directives to reflect the configuration of the SSL directives in the old server.
 - stopping the old server and starting the new one with your old site files and modified server configuration file in place.

To upgrade from Stronghold 2.3 or earlier

1. Install Stronghold 3.0 into a new directory, using the instructions in [Installing Stronghold Web Server on page 7](#).
2. Copy the contents of the old `<ServerRoot>/cgi-bin/` directory into the `Stronghold 3.0 cgi-bin/` directory.
3. Copy the contents of the old `<ServerRoot>/htdocs/` directory into the `Stronghold 3.0 htdocs/` directory. Take care not to overwrite Stronghold 3.0 documentation files and please note that PHP2 files are not supported by PHP3.
4. Copy the contents of the old `<ServerRoot>/conf/` directory into the `Stronghold 3.0 conf/` directory. Take care not to overwrite Stronghold 3.0 files.
5. Copy the contents of the old `<ServerRoot>/ssl/certs`, `<ServerRoot>/ssl/private` and `<ServerRoot>/ssl/CA` directories into the Stronghold 3.0 `ssl/certs`, `ssl/private`, `ssl/CA` directories.
6. If you use Stronghold's SWISH site indexer, copy any site indexes from the old `<ServerRoot>/swish/` directory into the `Stronghold 3.0 swish/` directory.
7. Create a new subdirectory in the `Stronghold 3.0 logs/` directory for storing your old log files:

```
# mkdir <ServerRoot>/logs/old
```

8. Copy the contents of the old `<ServerRoot>/logs/` directory into the new Stronghold 3.0 `<ServerRoot>/logs/old/` directory.
9. Open both the old and new server configuration files using a text editor in separate windows.
10. Copy and paste any custom configurations from the old file into the new file.

Using the new configuration file as a base, you must modify your custom configurations to comply with new configuration conventions. You must also configure Stronghold 3.0 SSL directives to reflect setup of your SSL directives in your old server. Please note that you may still use the old Stronghold 2.3 SSL directives if the option `CompatEnvVars` is set on for the new `SSLOptions` directive. Otherwise it is recommended that you use the

new Stronghold 3.0 SSL directives instead. Please refer to Chapter 7 of the Administration Guide for the new SSL directives.

11. Locate any `AuthName` directives and place quotes around their values. For example,

```
AuthName Marx Bros
```

should be changed to

```
AuthName "Marx Bros"
```

12. Edit the `<ServerRoot>` directive to reflect the path to the new Stronghold 3.0 directory.
13. Change the `StrongholdKey` directive to `StrongholdLicenseFile` and replace the old license key value with the name of the file that contains the server license block.
14. For every name-based virtual host configuration, enter a `NameVirtualHost` directive in the global configuration:

```
NameVirtualHost 209.60.53.41:80
```

```
NameVirtualHost 207.57.43.92:80
```

15. For every `<Files>` container, replace the full path to the file with the filename.
16. In previous versions, `<Files>` containers could stand alone and specify full paths to files. `<Files>` containers must now specify filenames instead of paths, but can also take wildcards and regular expressions. For example,

```
<Files /usr/local/www/htdocs/aquabats/batty.html>  
...  
</Files>
```

should be changed to:

```
<Directory /usr/local/www/htdocs/aquabats>  
<Files batty.html>  
...  
</Files>  
</Directory>
```

17. For every container that uses wildcards, modify the wildcard statement so that all slashes are literal.

Wildcards no longer match slash (/). For example,

```
<Directory /usr/local/www/htdocs/beavis*>
```

no longer matches any subdirectories of the `beavis` directory. Instead, it should be changed to

```
<Directory /usr/local/www/htdocs/beavis/*>
```

to match subdirectories one level below `beavis`, or

```
<Directory /usr/local/www/htdocs/beavis/**>
```

to match subdirectories two levels below.

18. Save the modified server configuration file.

19. Stop the old server

20. Start the new server with the `<ServerRoot>/bin/start-server` script.

Upgrading From Stronghold 2.4

Upgrading to Stronghold Web Server 3.0 from Stronghold 2.4 involves:

- copying all customized files from the old server directory tree to the new Stronghold 3.0 directory tree.
- adding custom configurations from old server configuration file to new server configuration file
- configuring Stronghold 3.0 SSL directives to reflect the configuration of the SSL directives in the old server
- stopping the old server and starting the new one with your old site files and modified server configuration file in place.

To upgrade from Stronghold 2.4

1. Install Stronghold Stronghold 3.0 into a new directory, using the instructions in [Installing Stronghold Web Server](#) on page 7.
2. Copy the contents of the old `<ServerRoot>/cgi-bin/` directory into the `Stronghold 3.0 cgi-bin/` directory.
3. Copy the contents of the old `<ServerRoot>/htdocs/` directory into the `Stronghold 3.0 htdocs/` directory. Take care not to overwrite Stronghold 3.0 documentation files and please note that PHP2 files are not supported by PHP3.

4. Copy the contents of the old `<ServerRoot>/conf/` directory into the `Stronghold 3.0 conf/` directory. Take care not to overwrite Stronghold 3.0. files.
5. Copy the contents of the old `<ServerRoot>/ssl/certs`, `<ServerRoot>/ssl/private` and `<ServerRoot>/ssl/CA` directories into the `Stronghold 3.0 ssl/certs`, `ssl/private`, `ssl/CA` directories.
6. If you use Stronghold's SWISH site indexer, copy any site indexes from the old `<ServerRoot>/swish/` directory into the `Stronghold 3.0 swish/` directory.
7. Create a new subdirectory in the `Stronghold 3.0 logs/` directory for storing the old log files:

```
#mkdir <ServerRoot>/logs/old
```

8. Copy the contents of the old `<ServerRoot>/logs/` directory into the new `Stronghold 3.0 <ServerRoot>/logs/old/` directory.
9. Open both the old and new server configuration files using a text editor in separate windows.
10. Copy and paste any custom configurations from the old file into the new file.
11. Compare the SSL directives in both old and new server configuration files. You must configure Stronghold 3.0 SSL directives to reflect setup of your SSL directives in your old server. Please note that you may still use the old Stronghold 2.4 SSL directives if the option `CompatEnvVars` is set on for the new `SSLOptions` directive. Otherwise it is recommended that you use the new Stronghold 3.0 SSL directives instead. Please refer to Chapter 7 of the Administration Guide for the new SSL directives.
12. Save the modified configuration file.
13. Stop the old server
14. Start the new server with the `<ServerRoot>/bin/start-server` script.

Upgrading from Apache

Upgrading to Stronghold Web Server 3.0 from an Apache Web Server 1.3.12 involves:

- copying all customized files from the old server directory tree to the new Stronghold 3.0 directory tree.
- adding custom configurations from old server configuration file to new server configuration file.
- modifying the new server configuration file (custom sections) to incorporate SSL support where necessary
- stopping the old server and starting the new one with your old site files and modified server configuration file in place.

To upgrade from an Apache Web Server

1. Install Stronghold Stronghold 3.0 into a new directory, using the instructions in [Installing Stronghold Web Server o page 7](#).
2. Copy the contents of the old `<ServerRoot>/cgi-bin/` directory into the [Stronghold 3.0 cgi-bin/](#) directory.
3. Copy the contents of the old `<ServerRoot>/htdocs/` directory into the [Stronghold 3.0 htdocs/](#) directory. Take care not to overwrite Stronghold 3.0 documentation files.
4. Copy the contents of the old `<ServerRoot>/conf/` directory into the [Stronghold 3.0 conf/](#) directory. Take care not to overwrite Stronghold 3.0 files.
5. If you use the SWISH site indexer, copy any site indexes from the old `swish/` directory into the [Stronghold 3.0 swish/](#) directory.
6. Create a new subdirectory in the [Stronghold 3.0 logs/](#) directory for storing your old log files:

```
# mkdir <ServerRoot>/logs/old
```
7. Copy the contents of the old `<ServerRoot>/logs/` directory into the new [Stronghold 3.0 <ServerRoot>/logs/old/](#) directory.
8. Open both the old and new server configuration files using a text editor in separate windows.
9. Copy and paste any custom configurations from the old file into the new file.

To configure additional virtual hosts as SSL virtual hosts, add SSL directives to each additional `<VirtualHost>` section. Please refer to Chapter 7 of the Administration Guide for the new SSL directives.

10. Save the modified configuration file.
11. Stop the old Apache server
12. Run the `<ServerRoot>/bin/start-server` script to start the new, SSL-enabled Stronghold server

Requesting and Installing New License Blocks

When you first install Stronghold Web Server 3.0, the installation program prompts you for your license block and installs it automatically. If you installed an evaluation version of Stronghold before you purchased it, you installed a temporary license block that expires when your evaluation period ends. You must install your permanent license block when you receive it. You can do this manually or by using the `install_lb` utility located in the `<ServerRoot>/bin/` directory.

License keys from Stronghold version 2.3 and below do not work with Stronghold 3.0
--

To install a license block with `install_lb`

13. Run the `install_lb` utility:

```
# <ServerRoot>/bin/install_lb
```

The utility prompts you for the license block.

14. Paste the license block into the `install_lb` window.

15. Press Control-D at the beginning of the first blank line after the license block. Press Return to get a blank line if none was inserted when you pasted the license block.

If the license block you pasted is not valid or there was no blank line when you pressed Control-D, the utility warns you of this and asks you if you want to try pasting the license block again. If the license block was pasted successfully, the utility exits and the license block is installed.

To install a license block manually

1. Back up the original `sh3licence` file:

```
# cp sh3licence sh2licence-backup
```

If the `StrongholdLicenseFile` directive in your server configuration specifies a different file, use that file instead.

2. Use a text editor to open the `<ServerRoot>/sh3licence` file.

3. Delete your previous license block.
4. Paste your new license block into the file.

Your license block should look something like this:

```
*****BEGIN LICENSE BLOCK*****
TEIAAQCKAAAAAGar2a7Pc0iD2oyGDN9a5mrYkZ0NrG7Zcy7UFbFoP4xLzvcVAN4K
0Mrww4z0A2o+gfVADbDz9IdePDKOA21C2E8SiPu1qwVI1pwvbA6xuVRWDo05BT/I
rNGExtSx+LMh3N1q2icj4eD53kNQVoLvaoQ5CcYvWGexrKiSDQvE3agLU2VyaWFs
OiAzMTQxNTkyNjUzNTg5NzkzMjMKQ29tcGFueTogQzJOZXQgRXVybSchmunkZApF
eHBpcmVzOiBOZXZlcgpQcm9kdWN0OiBTSDIKVHlwZfogRXZhbnVhdGlvbGpJREVB
OiBZZXMKRmFrZTogCklyb2t1biBMaW5lCklyb2t1bjogMQ==
*****END LICENSE BLOCK*****
```

5. Save the modified file.
6. Restart the server:

```
# <Serverroot>/bin/reload-server
```

Default Configuration

This appendix shows you Stronghold's default configuration. This is the [httpd.conf](#) file that you start with, and it includes everything you need to run a basic Web site. Your actual default configuration file is slightly different, depending on the options you choose during the installation process. The file also includes comments that explain each group of directives. Modify this file to customize the server.

##

```
## httpd.conf -- Apache HTTP server configuration file
##

#
# Based upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file. It contains the
# configuration directives that give the server its instructions.
# See <URL:http://www.apache.org/docs/> for detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do. They're here only as hints or reminders. If you are unsure
# consult the online docs. You have been warned.
#
# After this file is processed, the server will look for and process
# /home/build/sh3_install//conf/srm.conf and then /home/build/sh3_install//conf/
# access.conf
# unless you have overridden these with ResourceConfig and/or
# AccessConfig directives here.
#
# The configuration directives are grouped into three basic sections:
# 1. Directives that control the operation of the Apache server process as a
#    whole (the 'global environment').
# 2. Directives that define the parameters of the 'main' or 'default' server,
#    which responds to requests that aren't handled by a virtual host.
#    These directives also provide default values for the settings
#    of all virtual hosts.
# 3. Settings for virtual hosts, which allow Web requests to be sent to
#    different IP addresses or hostnames and have them handled by the
#    same Apache server process.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path. If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "logs/foo.log"
# with ServerRoot set to "/usr/local/apache" will be interpreted by the
# server as "/usr/local/apache/logs/foo.log".
#

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#
```

```
#
# File containing the Stronghold license block, relative to
# ServerRoot
#
StrongholdLicenseFile sh3license

#
# Use "StrongholdAccelerator" to enable any supported hardware
# accelerators. NB: If you enable an accelerator and the server
# does not start, consult the error logs and ensure your accelerator
# is functioning properly.
#
#StrongholdAccelerator cswift
StrongholdAccelerator none

#
# ServerType is either inetd, or standalone. Inetd mode is only supported on
# Unix platforms.
#
ServerType standalone

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE! If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation
# (available at <URL:http://www.apache.org/docs/mod/core.html#lockfile>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "/home/build/sh3_install/"

#
# The LockFile directive sets the path to the lockfile used when Apache
# is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or
# USE_FLOCK_SERIALIZED_ACCEPT. This directive should normally be left at
# its default value. The main reason for changing it is if the logs
# directory is NFS mounted, since the lockfile MUST BE STORED ON A LOCAL
# DISK. The PID of the main server process is automatically appended to
# the filename.
#
#LockFile logs/httpd.lock
```

Default Configuration

```
#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile logs/httpd.pid

#
# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this. But if yours does (you'll know because
# this file will be created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.
#
ScoreBoardFile logs/httpd.scoreboard

#
# In the standard configuration, the server will process this file,
# srm.conf, and access.conf in that order. The latter two files are
# now distributed empty, as it is recommended that all directives
# be kept in a single file for simplicity. The commented-out values
# below are the built-in defaults. You can have the server ignore
# these files altogether by using "/dev/null" (for Unix) or
# "nul" (for Win32) for the arguments to the directives.
#
ResourceConfig /dev/null
AccessConfig /dev/null

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
```

```
# same client on the same connection.
#
KeepAliveTimeout 15

#
# Server-pool size regulation.  Rather than making you guess how many
# server processes you need, Apache dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).
#
# It does this by periodically checking how many servers are waiting
# for a request.  If there are fewer than MinSpareServers, it creates
# a new spare.  If there are more than MaxSpareServers, some of the
# spares die off.  The default values are probably OK for most sites.
#
MinSpareServers 5
MaxSpareServers 10

#
# Number of servers to start initially --- should be a reasonable ballpark
# figure.
#
StartServers 5

#
# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# the system with it as it spirals down...
#
MaxClients 150

#
# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.  The child will exit so
# as to avoid problems after prolonged use when Apache (and maybe the
# libraries it uses) leak memory or other resources.  On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries.  For these platforms, set to something like 10000
# or so; a setting of 0 means unlimited.
#
# NOTE: This value does not include keepalive requests after the initial
# request per connection.  For example, if a child process handles
```

Default Configuration

```
#      an initial request and 10 subsequent "keptalive" requests, it
#      would only count as 1 request towards this limit.
#
MaxRequestsPerChild 10000

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
#Listen 3000
#Listen 12.34.56.78:80

#
# BindAddress: You can support virtual hosts with this option. This directive
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the <VirtualHost> and Listen directives.
#
#BindAddress *

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding 'LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Please read the file README.DSO in the Apache 1.3 distribution for more
# details about the DSO mechanism and run 'httpd -l' for the list of already
# built-in (statically linked and thus always available) modules in your httpd
# binary.
#
# Note: The order is which modules are loaded is important. Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so
LoadModule php3_module modules/libexec/libphp3.so

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information (ExtendedStatus
# Off) when the "server-status" handler is called. The default is Off.
#
ExtendedStatus On
```

```
### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition.  These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#
#
# If your ServerType directive (set earlier in the 'Global Environment'
# section) is set to "inetd", the next few directives don't have any
# effect since their settings are defined by the inetd configuration.
# Skip ahead to the ServerAdmin directive.
#
#
# Port: The port to which the standalone server listens. For
# ports < 1023, you will need httpd to be run as root initially.
#
Port 9001

##
## SSL Support
##
## When we also provide SSL we have to listen to the
## standard HTTP port (see above) and to the HTTPS port
##
Listen 9001
Listen 9002

#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
# . On SCO (ODT 3) use "User nouser" and "Group nogroup".
# . On HP-UX you may not be able to use shared memory as nobody, and the
#   suggested workaround is to create a user www and use that user.
# NOTE that some kernels refuse to setgid(Group) or semctl(IPC_SET)
# when the value of (unsigned)Group is above 60000;
# don't use Group nobody on these systems!
#
```

Default Configuration

```
User build
Group build

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin webmaster@loretta.localnet

#
# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
ServerName loretta.localnet

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/home/build/sh3_install//htdocs"

#
# Each directory to which Apache has access, can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# permissions.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

#
# Note that from this point forward you must specifically allow
```

```
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/home/build/sh3_install//htdocs">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
    Options Indexes FollowSymLinks MultiViews

#
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all
</Directory>

#
# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is received.
#
UserDir public_html

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
#<Directory /home/*/public_html>
#     AllowOverride FileInfo AuthConfig Limit
#     Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
```

Default Configuration

```
# <Limit GET POST OPTIONS PROPFIND>
#     Order allow,deny
#     Allow from all
# </Limit>
# <LimitExcept GET POST OPTIONS PROPFIND>
#     Order deny,allow
#     Deny from all
# </LimitExcept>
#</Directory>

#
# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index.  Separate multiple entries with spaces.
#
DirectoryIndex index.html index.htm index.php index.php3 index.cgi

#
# AccessFileName: The name of the file to look for in each directory
# for access control information.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess files from being viewed by
# Web clients.  Since .htaccess files often contain authorization
# information, access is disallowed for security reasons.  Comment
# these lines out if you want Web visitors to see the contents of
# .htaccess files.  If you change the AccessFileName directive above,
# be sure to make the corresponding changes here.
#
# Also, folks tend to use names such as .htpasswd for password
# files, so this will protect those as well.
#
<Files ~ "\.ht">
    Order allow,deny
    Deny from all
</Files>

#
# CacheNegotiatedDocs: By default, Apache sends "Pragma: no-cache" with each
# document that was negotiated on the basis of content.  This asks proxy
# servers not to cache the document.  Uncommenting the following line disables
# this behavior, and proxies will be allowed to cache the documents.
#
#CacheNegotiatedDocs
```

```
#
# UseCanonicalName: (new for 1.3) With this setting turned on, whenever
# Apache needs to construct a self-referencing URL (a URL that refers back
# to the server the response is coming from) it will use ServerName and
# Port to form a "canonical" name. With this setting off, Apache will
# use the hostname:port that the client supplied, when possible. This
# also affects SERVER_NAME and SERVER_PORT in CGI scripts.
#
UseCanonicalName On

#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
TypesConfig /home/build/sh3_install//conf/mime.types

#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value. If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type. The MIMEMagicFile
# directive tells the module where the hint definitions are located.
# mod_mime_magic is not part of the default server (you have to add
# it yourself with a LoadModule [see the DSO paragraph in the 'Global
# Environment' section], or recompile the server and include mod_mime_magic
# as part of the configuration), so it's enclosed in an <IfModule> container.
# This means that the MIMEMagicFile directive will only be processed if the
# module is part of the server.
#
<IfModule mod_mime_magic.c>
    MIMEMagicFile /home/build/sh3_install//conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
```

Default Configuration

```
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here.  Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
CustomLog logs/access_log common

#
# If you would like to have agent and referer logfiles, uncomment the
# following directives.
#
#CustomLog logs/referer_log referer
#CustomLog logs/agent_log agent
```

```
#
# If you prefer a single logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
#CustomLog logs/access_log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of: On | Off | EMail
#
ServerSignature On

#
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL.  So "/icons" isn't aliased in this
# example, only "/icons/"..
#
Alias /icons/ "/home/build/sh3_install//icons/"

<Directory "/home/build/sh3_install//icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ "/home/build/sh3_install//cgi-bin/"

#
# "/home/build/sh3_install//cgi-bin" should be changed to whatever your
ScriptAliased
```

Default Configuration

```
# CGI directory exists, if you have that configured.
#
<Directory "/home/build/sh3_install//cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

#
# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect old-URI new-URL
#
#
# Directives controlling the display of server-generated directory listings.
#
#
# FancyIndexing is whether you want fancy directory indexing or standard
#
IndexOptions FancyIndexing

#
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions. These are only displayed for
# FancyIndexed directories.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrm .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
```

```
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file in
# server-generated indexes. These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName is the name of the README file the server will look for by
# default, and append to directory listings.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
#
# The server will first look for name.html and include it if found.
# If name.html doesn't exist, the server will then look for name.txt
# and include it as plaintext if found.
#
ReadmeName README
HeaderName HEADER

#
# IndexIgnore is a set of filenames which directory indexing should ignore
# and not include in the listing. Shell-style wildcarding is permitted.
```

```
#
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

#
# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have nothing
# to do with the FancyIndexing customization directives above.
#
AddEncoding x-compress Z
AddEncoding x-gzip gz tgz

#
# AddLanguage allows you to specify the language of a document. You can
# then use content negotiation to give a browser a file in a language
# it can understand.
#
# Note 1: The suffix does not have to be the same as the language
# keyword --- those with documents in Polish (whose net-standard
# language code is pl) may wish to use "AddLanguage pl .po" to
# avoid the ambiguity with the common suffix for perl scripts.
#
# Note 2: The example entries below illustrate that in quite
# some cases the two character 'Language' abbreviation is not
# identical to the two character 'Country' code for it's country,
# E.g. 'Danmark/dk' versus 'Danish/da'.
#
# Note 3: In the case of 'ltz' we violate the RFC by using a three char
# specifier. But there is 'work in progress' to fix this and get
# the reference data for rfc1766 cleaned up.
#
# Danish (da) - Dutch (nl) - English (en) - Estonian (ee)
# French (fr) - German (de) - Greek-Modern (el)
# Italian (it) -Portugese (pt) - Luxembourgeois* (ltz)
# Spanish (es) - Swedish (sv) - Catalan (ca) - Czech(cz)
#
AddLanguage da .dk
AddLanguage nl .nl
AddLanguage en .en
AddLanguage et .ee
AddLanguage fr .fr
AddLanguage de .de
AddLanguage el .el
AddLanguage it .it
AddLanguage pt .pt
AddLanguage ltz .lu
```

```
AddLanguage ca .ca
AddLanguage es .es
AddLanguage sv .se
AddLanguage cz .cz

# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
#
# Just list the languages in decreasing order of preference. We have
# more or less alphabetized them here. You probably want to change this.
#
LanguagePriority en da nl et fr de el it pt ltz ca es sv

#
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP 3.x module (not part of the Apache distribution - see
# http://www.php.net) will typically use:
#
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source .phps
#
# And for PHP 4.x, use:
#
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps

<IfModule mod_php3.c>
AddType application/x-httpd-php3 .php3
AddType application/x-httpd-php3-source .phps
</IfModule>

<IfModule mod_php4.c>
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
</IfModule>

AddType application/x-tar .tgz

#
# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
#
# If you want to use server side includes, or CGI outside
```

Default Configuration

```
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi

#
# To use server-parsed HTML files
#
#AddType text/html .shtml
#AddHandler server-parsed .shtml

#
# Uncomment the following line to enable Apache's send-asis HTTP file
# feature
#
#AddHandler send-as-is asis

#
# If you wish to use server-parsed imagemap files, use
#
#AddHandler imap-file map

#
# To enable type maps, you might want to use
#
#AddHandler type-map var

#
# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# MetaDir: specifies the name of the directory in which Apache can find
# meta information files. These files contain additional HTTP headers
# to include when sending the document
#
#MetaDir .web

#
# MetaSuffix: specifies the file name suffix for the file containing the
# meta information.
```

```
#
#MetaSuffix .meta

#
# Customizable error response (Apache style)
# these come in three flavors
#
# 1) plain text
#ErrorDocument 500 "The server made a boo boo.
# n.b. the (" marks it as text, it does not get output
#
# 2) local redirects
#ErrorDocument 404 /missing.html
# to redirect to local URL /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
# N.B.: You can redirect to a script or a document using server-side-includes.
#
# 3) external redirects
#ErrorDocument 402 http://some.other_server.com/subscription_info.html
# N.B.: Many of the environment variables associated with the original
# request will *not* be available to such a script.

#
# The following directives modify normal HTTP response behavior.
# The first directive disables keepalive for Netscape 2.x and browsers that
# spoof it. There are known problems with these browser implementations.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

#
# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
#
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

#
# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your_domain.com" to match your domain to enable.
#
```

```
#<Location /server-status>
#   SetHandler server-status
#   Order deny,allow
#   Deny from all
#   Allow from .your_domain.com
#</Location>

#
# Allow remote server configuration reports, with the URL of
# http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".your_domain.com" to match your domain to enable.
#
#<Location /server-info>
#   SetHandler server-info
#   Order deny,allow
#   Deny from all
#   Allow from .your_domain.com
#</Location>

<Location /stronghold-status>
SetHandler server-status
</Location>

<Location /stronghold-info>
SetHandler server-info
</Location>

#
# There have been reports of people trying to abuse an old bug from pre-1.1
# days.  This bug involved a CGI script distributed as a part of Apache.
# By uncommenting these lines you can redirect these attacks to a logging
# script on phf.apache.org.  Or, you can record them yourself, using the script
# support/phf_abuse_log.cgi.
#
#<Location /cgi-bin/phf*>
#   Deny from all
#   ErrorDocument 403 http://phf.apache.org/phf_abuse_log.cgi
#</Location>

#
# Proxy Server directives. Uncomment the following lines to
# enable the proxy server:
#
#<IfModule mod_proxy.c>
#ProxyRequests On
#
```

```
#<Directory proxy:*>
#   Order deny,allow
#   Deny from all
#   Allow from .your_domain.com
#</Directory>

#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via: headers)
# Set to one of: Off | On | Full | Block
#
#ProxyVia On

#
# To enable the cache as well, edit and uncomment the following lines:
# (no cacheing without CacheRoot)
#
#CacheRoot "/home/build/sh3_install//proxy"
#CacheSize 5
#CacheGcInterval 4
#CacheMaxExpire 24
#CacheLastModifiedFactor 0.1
#CacheDefaultExpire 1
#NoCache a_domain.com another_domain.edu joes.garage_sale.com

#</IfModule>
# End of proxy directives.

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them.
# Please see the documentation at <URL:http://www.apache.org/docs/vhosts/>
# for further details before you try to setup virtual hosts.
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# If you want to use name-based virtual hosts you need to define at
# least one IP address (and port number) for them.
#
#NameVirtualHost 12.34.56.78:80
#NameVirtualHost 12.34.56.78

#
# VirtualHost example:
```

Default Configuration

```
# Almost any Apache directive may go into a VirtualHost container.
#
#<VirtualHost ip.address.of.host.some_domain.com>
#   ServerAdmin webmaster@host.some_domain.com
#   DocumentRoot /www/docs/host.some_domain.com
#   ServerName host.some_domain.com
#   ErrorLog logs/host.some_domain.com-error_log
#   CustomLog logs/host.some_domain.com-access_log common
#</VirtualHost>

#<VirtualHost _default_:*>
#</VirtualHost>

##
## SSL Global Context
##
## All SSL configuration in this context applies both to
## the main server and all SSL-enabled virtual hosts.
##

#
# Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl

<IfModule mod_ssl.c>

# Pass Phrase Dialog:
# Configure the pass phrase gathering process.
# The filtering dialog program ('builtin' is a internal
# terminal dialog) has to provide the pass phrase on stdout.
SSLPassPhraseDialog builtin

# Inter-Process Session Cache:
# Configure the SSL Session Cache: First either 'none'
# or 'dbm:/path/to/file' for the mechanism to use and
# second the expiring timeout (in seconds).
#SSLSessionCache         none
#SSLSessionCache         dbm:logs/ssl/ssl_scache
SSLSessionCache         shm:logs/ssl/ssl_scache(512000)
SSLSessionCacheTimeout  300

# Semaphore:
# Configure the path to the mutual exclusion semaphore the
# SSL engine uses internally for inter-process synchronization.
```

```
SSLMutex file:logs/ssl/ssl_mutex

# Pseudo Random Number Generator (PRNG):
# Configure one or more sources to seed the PRNG of the
# SSL library. The seed data should be of good random quality.
# WARNING! On some platforms /dev/random blocks if not enough entropy
# is available. This means you then cannot use the /dev/random device
# because it would lead to very long connection times (as long as
# it requires to make more entropy available). But usually those
# platforms additionally provide a /dev/urandom device which doesn't
# block. So, if available, use this one instead. Read the mod_ssl User
# Manual for more details.
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
#SSLRandomSeed startup file:/dev/random 512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random 512
#SSLRandomSeed connect file:/dev/urandom 512

# Logging:
# The home of the dedicated SSL protocol logfile. Errors are
# additionally duplicated in the general error log file. Put
# this somewhere where it cannot be used for symlink attacks on
# a real server (i.e. somewhere where only root can write).
# Log levels are (ascending order: higher ones include lower ones):
# none, error, warn, info, trace, debug.
SSLLog logs/ssl/ssl_engine_log
SSLLogLevel warn

</IfModule>

##
## SSL Virtual Host Context
##

<VirtualHost _default_:9002>

# General setup for the virtual host
DocumentRoot "/home/build/sh3_install//htdocs"
ServerName loretta.localnet
ServerAdmin webmaster@loretta.localnet
ErrorLog logs/ssl/error_log
TransferLog logs/ssl/access_log

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
```

Default Configuration

```
SSLEngine on

# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate.
# See the mod_ssl documentation for a complete list.
#SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. A test
# certificate can be generated with 'make certificate' under
# built time. Keep in mind that if you've both a RSA and a DSA
# certificate you can configure both in parallel (to also allow
# the use of DSA ciphers, etc.)
SSLCertificateFile /home/build/sh3_install//ssl/certs/loretta.localnet.cert
#SSLCertificateFile /home/build/sh3_install//ssl/certs/loretta.localnet-dsa.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /home/build/sh3_install//ssl/private/loretta.localnet.key
#SSLCertificateKeyFile /home/build/sh3_install//ssl/private/loretta.localnet-
dsa.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /home/build/sh3_install//ssl/certs/
loretta.localnet.chain

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
#SSLCACertificatePath /home/build/sh3_install//conf/ssl.crt
SSLCACertificateFile /home/build/sh3_install//ssl/CA/client-rootcerts.pem
```

```

# Certificate Revocation Lists (CRL):
# Set the CA revocation path where to find CA CRLs for client
# authentication or alternatively one huge file containing all
# of them (file must be PEM encoded)
# Note: Inside SSLCARevocationPath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /home/build/sh3_install//conf/ssl.crl
#SSLCARevocationFile /home/build/sh3_install//ssl/certs/ca-bundle.crl

# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10

# Access Control:
# With SSLRequire you can do per-directory access control based
# on arbitrary complex boolean expressions containing server
# variable checks and other lookup directives. The syntax is a
# mixture between C and Perl. See the mod_ssl documentation
# for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
#               and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#               and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#               and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#               and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20          ) \
#               or %{REMOTE_ADDR} =~ m/^192\.76\.162\.([0-9]+$/
#</Location>

# SSL Engine Options:
# Set various options for the SSL engine.
# o FakeBasicAuth:
#   Translate the client X.509 into a Basic Authorisation. This means that
#   the standard Auth/DBMAuth methods can be used for access control. The
#   user name is the 'one line' version of the client's X.509 certificate.
#   Note that no password is obtained from the user. Every entry in the user
#   file needs this password: 'xxj31ZMTZzkVA'.
# o ExportCertData:
#   This exports two additional environment variables: SSL_CLIENT_CERT and
#   SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#   server (always existing) and the client (only existing when client

```

```
# authentication is used). This can be used to import the certificates
# into CGI scripts.
# o StdEnvVars:
# This exports the standard SSL/TLS related 'SSL_*' environment variables.
# Per default this exportation is switched off for performance reasons,
# because the extraction step is an expensive operation and is usually
# useless for serving static content. So one usually enables the
# exportation for CGI and SSI requests only.
# o CompatEnvVars:
# This exports obsolete environment variables for backward compatibility
# to Apache-SSL 1.x, mod_ssl 2.0.x, Sioux 1.0 and Stronghold 2.x. Use this
# to provide compatibility to existing CGI scripts.
# o StrictRequire:
# This denies access when "SSLRequireSSL" or "SSLRequire" applied even
# under a "Satisfy any" situation, i.e. when it applies access is denied
# and no other module can change it.
# o OptRenegotiate:
# This enables optimized SSL connection renegotiation handling when SSL
# directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars +StrictRequire
<Files ~ "\.(cgi|shtml)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/home/build/sh3_install//cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

SSLOptions +CompatEnvVars

# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant shutdown
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
# This forces an unclean shutdown when the connection is closed, i.e. no
# SSL close notify alert is send or allowed to received. This violates
# the SSL/TLS standard but is needed for some brain-dead browsers. Use
# this when you receive I/O errors because of the standard approach where
# mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
# This forces an accurate shutdown when the connection is closed, i.e. a
# SSL close notify alert is send and mod_ssl waits for the close notify
# alert of the client. This is 100% SSL/TLS standard compliant, but in
# practice often causes hanging connections with brain-dead browsers. Use
# this only for browsers where you know that their SSL implementation
```

```
#     works correctly.
#     Notice: Most problems of broken clients are also related to the HTTP
#     keep-alive facility, so you usually additionally want to disable
#     keep-alive for those clients, too. Use variable "nokeepalive" for this.
SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown

#     Per-Server Logging:
#     The home of a custom SSL log file. Use this when you want a
#     compact non-error SSL logfile on a virtual host basis.
CustomLog logs/ssl/ssl_request_log \
    "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

Glossary

[A](#) | [C](#) | [D](#) | [E](#) | [F](#) | [H](#) | [I](#) | [K](#) | [L](#) | [M](#) | [P](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | [X](#)

A

absolute path	A path that begins with a slash (/). It indicates that the specified path is relative to the top-level directory on the host system.
access path	The restriction of access to network realms, usually by permitting access by a discrete set of hosts or users and denying it to all others.
algorithm	An unambiguous formula or set of rules of solving a finite number of steps. Algorithms for encryption are called <i>ciphers</i> .
Apache	A free UNIX Web server which forms the core of the Stronghold server. See Apache HTTPD Server Project site at http://www.apache.org/httpd.html
authentication	The positive identification of a network entity as a server, a client or a user.

C

CA certificate	A certificate containing a Certification Authority's (CA) public key. Network entities use this public key to verify certificates signed with the CA's private key.
certificate	A file used for authenticating network entities under the SSL protocol. A certificate contains information about its owner (called the subject) and its issuer, plus the owner's public key and a signature made by a Certification Authority (CA). Network entities verify these signatures using CA certificates.
certificate signing request (CSR)	An unsigned certificate for submission to a Certification Authority that signs it with its private key. Once the CSR is signed, it becomes a certificate.
Certification Authority (CA)	A trusted third party whose purpose it is to sign certificates for network entities it has authenticated using secure means. Other network entities can check certificate signatures to verify that a CA has authenticated the bearer of a certificate.
child process	A subordinate process started by a parent process. Stronghold creates child processes to answer incoming requests simultaneously.
cipher	A system of encryption.
ciphertext	Encrypted data.
client certificate	A certificate authenticating a network client and signed by a Certification Authority (CA). It contains the client's public key.
comment out	To prefix one or more lines with a hash mark (#), marking it for omission from processing, compilation, and interpretation. When Stronghold reads a file, it skips any lines that are commented out.
Common Gateway Interface (CGI)	A standard interface between a Web server and other programs.
configuration file	The <code><ServerRoot>/conf/httpd.conf</code> file that contains the parameters to control Stronghold's behavior.

CONNECT	A proxying protocol under TCP/IP. It can be used to encapsulate other protocols, such as the SSL proxy protocol.
container	A pair of configuration delimiters specifying an object on the server. Directives pertaining to the object go inside the container, between the opening and closing delimiters
content negotiation	The negotiation of file format or language between client and server. Clients and servers can rank formats and languages in order of preference, then negotiate for the most desirable version.
cryptanalysis	The analysis of ciphers for the purpose of evaluating their security, usually by attempting to break them
cryptography	The study and practice data encoding that provides techniques for protecting the secure and/or authenticity of data for its “author” and/or “recipient.”
D	
decryption	The conversion of encrypted information (ciphertext) to its original, readable form (plaintext).
denial-of-service attack	A network attack created by flooding a host with data. When a server’s resources are occupied by such an attack, legitimate requests may be denied. If the rate and volume of data reach a certain level, the server program may crash.
digest	The hash-value of a message, which can be cryptographically signed so that others can verify that the contents of the message have not been altered in transit.
directive	A configuration command that controls one or more aspects of a program’s behavior
Domain Naming System (DNS)	A distributed system for resolving fully-qualified domain names into IP numbers.

E

encryption	The encoding of data in order to hide its content from everyone except its intended viewer. Encryption converts readable data (plaintext) into unreadable encrypted data (ciphertext).
export-crippled	Diminished in cryptographic strength (and security) in order to comply with the United States' Export Administration Regulations (EAR). Export-crippled cryptographic software is limited to using small key sizes, resulting in ciphertext that can be decrypted by a brute force attack to find the encryption key.

F

firewall	A dedicated gateway computer that holds no sensitive data. The firewall intercepts and filters incoming data packets in order to control and sometimes log access to internal computer systems.
FTP	The File Transfer Protocol, a client-server protocol that allows a user to transfer files to and from a server over a TCP/IP network.
fully-qualified domain name	The unique name of a network entity, consisting of a hostname and a domain name that can resolve to an IP address. For example, "www" is a hostname, "c2.net" is a domain name, and "www.c2.net" is a fully-qualified domain name.

H

handler	An aspect of the server that handles requests, or certain classes of requests.
---------	--

hash	A smaller number produced using a deterministic function and an arbitrarily large input. Also known as a digest, it is cryptographically signed so that others can verify that the contents of the message have not been altered in transit.
host	A networked computer that provides services that other computers or terminals can access.
hostname	A unique name that identifies a single host within a network domain.
HTTP	The Hypertext Transport Protocol, the standard protocol of the World Wide Web.
httpd.conf	The Stronghold runtime configuration file, containing directives which the server reads on startup.
HTTPS	The Hypertext Transport Protocol, Secure, the standard SSL communication mechanism of the World Wide Web.

I

IP Address	A 32-bit, dotted decimal address, such as 204.17.233.16, that uniquely identifies a network domain.
------------	---

K

keepalive	A server feature which keeps a TCP connection open after an initial request in order to fulfill subsequent requests. keepalive's eliminate the server overhead normally required to open and close an individual connection for each request.
Key	A value that must be fed into a cryptographic algorithm in order to encrypt or decrypt a message.
key pair	A set of two keys used in public key cryptography. One is the public key used to encrypt the data or verify signed data. The other is the private key necessary to sign data or decrypt data that has been encrypted with the public key.

L

License key	A string of colon-separated, hexadecimal values that Stronghold WebServer reads on startup to verify that the server software is licensed for the host on which it runs.
local-area network (LAN)	A private group of computers and related peripheral devices that are connected and capable of sharing resources. A LAN may or may not be connected to the Internet.

M

make	The UNIX tool that compiles the Stronghold server and all its components binary based on instructions in the Makefile.
Makefile	A file containing instructions that tell make how to compile Stronghold, taking into account the interdependencies of modules and their modifications times.
metainformation	Information about information. Metainformation may include descriptive information about the size, format, or other characteristics of a client request, server response, or a document being transferred.
MIME	(Multipurpose Internet Mail Extensions). A standard for providing metainformation about documents on the Internet, including non-textual data. The main use of MIME is in identifying the type of data being exchanged and how it should be interpreted.

P

parent process	An original, controlling process i a group of related processes. The process it spawns are called <i>child processes</i> or children. The parent process in Stronghold does not answer browser requests, but creates and administers child processes that answer the request.
pass phrase	The word or phrase that is used to encrypt and decrypt a private key when it is stored on disk. The pass phrase prevents unauthorized users from starting, restarting, or reconfiguring the server, and protocols the key itself form exploitation.
permission	An access privilege associated with a file or directory, indicating who can use it and how.
plaintext	Readable, unencrypted data.
PHP: Hypertext Processor	Personal home Page, and HTML-embedded scripting language originally designed for simple home page development. PHP has evolved to become an extended scripting language capable of replacing for CGI and SSI.
platform	An operating system environment and the hardware that supports it.
private key	The secret key in a key pair, used to decrypt incoming messages and sign outgoing ones.
process	In a multitasking environment such as UNIX, one instance of an executing program. The original instance of a program is called theparent process, and it may start one or more child processes. Stronghold starts and maintains multiple processes for answering many requests simultaneously.
proxy server	A server that relays requests and responses between clients and other servers.
public key	The publicly available key in a key pair, used to encrypt messages bound for its owner and to decrypt signatures made by its owner. See key pair

public key cryptography The study and application of asymmetric encryption systems, which use one key for encryption or signature verification and another for decryption or signature generation. A corresponding pair of such keys constitutes a *key pair*.

R

relative path A partial path that does not begin with a slash (/). It is interpreted as relative to the current directory.

request A message sent from a client to a server requesting one or more server resources.

response A message sent from a server to a client in relation to a request. A response may contain the requested resources, or it may contain additional information pertaining to the request.

root The superuser of a UNIX system, a special user with unlimited access to all files, directories and commands.

S

Secure Sockets Layer (SSL) A protocol created by Netscape Communications Corporation for authentication and encryption over TCP/IP networks, including the Web.

<ServerRoot> The path to the top-level Stronghold directory. In this manual `<ServerRoot>` is always a variable; substitute the actual path.

server-side includes (SSIs) HTML-embedded commands executed by the server before sending the HTML file to the client.

session A series of two or more related transactions between a client and a server. A session ends when the client quits or the session identifier expires.

signature An encrypted text block that validates a certificate or other file. A Certification Authority (CA) creates a signature by generating a hash of a certificate request, then signing the hash with its own private key. Only the CA's public key can correctly verify the signature, and this verifies that the CA has authenticated the network entity that owns the signed certificate.

site certificate	A certificate authenticating a network host or virtual host that has been signed by a Certification Authority (CA). A site certificate contains the site's public key.
source	One or more text files, written in a programming language, which can be compiled to form one or more binary files.
SSLeay	An SSL library developed by Eric Young (eay@cryptsoft.com). This library was used as the base code for OpenSSL, a much larger project with a consortium of developers and companies involved in maintenance and ongoing development. OpenSSL is the library that provides Stronghold cryptographic functionality.
status code	A numerical code denoting the status of a client request response.
SWISH	Simple Web Indexing System for Humans, Stronghold's standalone site-indexing program, located in the <code><ServerRoot>/SWISH</code> directory.
symmetric cryptography	The study and application of ciphers that use a single key for both encryption and decryption operations.

T

TCP/IP	Transmission Control Protocol/Internet Protocol, the suite of standard Internet protocols upon which HTTP, HTTPS, FTP, TELENET, and GOPHER are based.
Transport Layer Security (TLS)	A protocol for authentication and encryption over TCP/IP networks, including the Web. TLS is the successor to SSL version 3.0 and is nearly identical. It is being standardizing by the Internet Engineering Task Force.

U

uncomment	To remove the hash marks (#) that comment out, or neutralize, one or more lines of text. Programs read a line within a file only if it is uncommented.
-----------	--

V

virtual host A domain that shares a host with other domains.

W

WWWWAIS Stronghold's stand-alone search gateway, which searches site index files created by SWISH.

X

x.509 An authentication certificate scheme recommended by the International Telegraph and Telephone Consultive Committee and used in SSL authentication.