Sun WebServer Command Reference

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043-1100
U.S.A.

Part No: 805-5735
Jun 27, 1998

Contents

CHAPTER

man Pages(1m): Maintenance Commands

NAME

httpd.intro – Introduction to the man pages of the Sun™ WebServer™ command-line utilities. The man pages offer detailed instructions and examples on options and subcommands for each utility.

DESCRIPTION

The command-line utilities are available to administer Sun WebServer.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtman
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/access.conf` A web site's Access Control Lists (ACLs) configuration file.

`/etc/http/access.conf` The server administration ACLs configuration file.

site_path`/conf/access.conf` A web site's ACLs configuration file.

`/etc/http/access.conf` Defines the realms used to define users and groups for server administration.

site_path`/conf/content.conf` Defines the content variants, encoding types, and directory preferences for a web site.

`/etc/http/httpd-instances.conf` Tracks all Sun WebServer instances. When `htserver` creates a new server instance, an entry is added to this file.

site_path`/conf/`*site_name*`.httpd.conf` Contains the web site servlet engine configuration if the servlet engine is not shared.

`/etc/http/`*instance_name*`.httpd.conf` The server instance configuration file. When `hthost` adds a new site, it creates an entry in `httpd.conf` to define the *site_path* and web site configuration file.

site_path`/conf/`*site_name*`.site.conf` The web site configuration file.

`/etc/http/httpd.conf` Contains the server instance servlet engine configuration if all web sites share the servlet engine.

site_path/conf/map.conf	Creates an alias to a path on the file system or a redirection to a remote URL from a Uniform Resource Identifier (URI) on the host.
site_path/conf/realms.conf	Defines realms of user and group information used by access control lists on a Sun WebServer web site.
site_path/conf/servlets.properties	Defines each servlet that can be loaded by a web site.

SEE ALSO

access.conf(4), **content.conf**(4), **httpd.conf**(4), **httpd.cgi.logs**(4), **httpd.event.logs**(4), **httpd-instances.conf**(4), **httpd.request.logs**(4), **httpd.servlet.logs**(4), **httpd.site.conf**(4), **map.conf**(4), **realms.conf**(4), **servlets.properties**(4)

NOTES

htaccess	Adds or deletes ACLs for resources on a web site. An ACL applies to any token that can be a URI on that site, whether that URI is a directory, file, servlet, CGI, or alias to another resource.
htcontent	Create or deletes information about the content of resources on a web site. For directories, you can set whether they are avaliable to browse and specify the format for directory listings. For files, you can set preferences for HTTP 1.1 content negotiation. Preferences include character set, language, compression encoding, and media type.
hthost	Associates a web site (or virtual host) with system resources, such as the server instance that hosts the site, a configuration directory, a configuration file, and a host name. Activates or shuts down web sites on a running server instance.
htmap	Adds and deletes aliases from one URI to another resource on a web site. Redirects a token to nonfile resources such as servlets, CGI scripts, or Sun WebServer Administration Console.
htpasswd	Changes passwords for users in HTPASSWD realms. It is provided as a tool that can be incorporated in CGI or other scripts to automate password maintenance. User must be created

	using `htrealm`. Once a user is created, any system user can run `htpasswd` to update passwords (as long as the realm administrator name and password are specified).
htrealm	Creates, deletes, and lists realm definitions for use with ACLs. It can also be used to manage users and groups in `HTPASSWD` realms.
htserver	Creates and maintains Sun WebServer server instances. Each server instance is a process associated with a configuration file, and each one hosts one or more web sites. `htserver` can start, stop, and restart server instances. It can also enable or disable server instances.
htservlet	Configures the behavior of a servlet engine. Defines and modifies servlet engine runtime, security, and logging properties; adds or removes entries in the `servlets.properties` file, and loads, reloads, or unloads servlets in running servlet engines.
httpd	Server instances can be started or stopped by using the `htserver` utility, through the Sun WebServer Administration Console, or executing this script. It is recommended that you use `htserver` or the Sun WebServer Administration Console.

If the command is run by `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

htaccess – Allows manipulation of Sun™ WebServer™ access control lists (ACLs) configuration.

SYNOPSIS

htaccess add [–a][–g *group_name*|–I *Internet_host*|–u *username*] –h *hostname* –i *instance* [–m *method*] [–r *realm_name*] [–s *scheme_name*] –U *URI_name* [– y|–n] [–z *admin*[–p]]

htaccess check [–a][–g *group_name*|–I *Internet_host*|–u *user_name*] –h *hostname* –i *instance* [–m *method*] –U *URI_name* [–z *admin*[–p]]

htaccess delete [–a][–g *group_name*|–I *Internet_host*|–u *user_name*] –h *hostname* –i *instance* [–m *method*] –r *realm_name* –s *scheme_name* –U *URI_name* [–y|–n] [–z *admin*[–p]]

htaccess help

htaccess list –h *hostname* –i *instance* –U *URI_name* [–z *admin*[–p]]

htaccess version

DESCRIPTION

`htaccess` adds or deletes access control lists (ACLs) for resources on a web site. An ACL applies to any token that can be a Uniform Resource Identifier (URI) on that site, whether that URI is a directory, file, servlet, CGI, or alias to another resource.

ACLs can restrict access by the host name or IP address of a client, or by authenticated user or group name. Users are authenticated against a named realm (see **`htrealm`**(1m)), which defines whether passwords are stored in a Sun WebServer specific file (HTPASSWD), the operating system (UNIXSYS), in the Solaris™ for ISPs™ environment, an LDAP directory of ISP subscribers (ISP), or in the Management Console for Solaris for ISPs (ISPADMIN).

The `list` subcommand can be used to list the current ACLs on a URI.

The `check` subcommand can be used to check whether a user name and password can access a URI.

OPTIONS

Subcommands

The following subcommands are supported:

add	Adds a new ACL or permission to an existing ACL.
check	Checks if the specified access is allowed.

delete	Deletes an ACL or permission to an existing ACL.
help	Displays help on usage.
list	Lists all ACLs and their permissions for a given URL or specified host.
version	Displays the version of htaccess.

Options

The following options are supported:

–a	Specifies that the user or group is the administrator. Valid with the add and delete subcommands.
–g ***group_name***	Specifies the group in the realms to which the permissions apply. A group is a group of users defined in the realm. Use the wild card * to indicate that the permission applies to any *group_name*. Valid with the add, delete, and check subcommands.
–h ***hostname***	Specifies the name of the virtual host containing the ACL. Valid with all subcommands.
–I ***Internet_host***	Specifies the IP or domain to which the permissions apply. *Internet_host* can be a fully qualified or partial domain name. If the domain name is partial, the permission applies to all hosts whose fully qualified names end with the domain. It can also be a fully qualified or partial IP address. If the IP address is partial, the permission applies to all hosts whose IP address begins with the *Internet_host*. Use the wild card * to indicate that the permission applies to any *Internet_host*. Valid with the add, delete, and check subcommands.
–i ***instance***	Specifies the name of the httpd instance. Valid with all subcommands.
–m ***method_names***	Specifies the HTTP method name to which the permissions apply. The *method_names* directive is a list of ALL, DELETE, GET, POST, and PUT. Select ALL to permit all HTTP methods. Separate multiple methods with a space. The default is ALL. Valid with the add, delete, and check subcommands.
–n	Denies access permission to the named user, group, or host. Valid with the add and delete subcommands.

–p	Turns off prompting of password such that passwords are taken in from stdin and scripts may pipe (|) passwords. Valid with all subcommands.
–r ***realm_name***	Specifies the realm name. Valid with the add subcommand.
–s ***scheme_name***	Specifies authentication scheme. Valid with the add subcommand.

BASIC	The server expects user name and password information in base64 encoded text.
MD5	The server expects user name and a message digest of the password. The server must get the password in base64 encoded text locally, create a message digest, and compare it to the digest sent by the client. Valid only with `htpasswd`.
NONE	The server does not expect any authentication.

–U ***URI_name***	Specifies URI name protected by the ACL. Valid with all subcommands.
–u ***user_name***	Specifies the user name to which the permission applies. A user is any user with a user name for which Sun WebServer retrieves the password from the realm name specified in the ACL. Use the wild card * to indicate that the permission applies to any user. Valid with the add, check, and delete subcommands.
–v	Specifies the verbose mode. Valid with all subcommands.
–y	Allows the named user, group, or host to access the URI. Valid with the add and delete subcommands.
–z ***admin_name***	Specifies the name of the administrator. Valid with the add, check, and delete subcommands.

EXAMPLES

EXAMPLE 1

To protect the URL `http://www.A.com/project/` on the server instance "sws_server" using an HTPASSWD realm "Project" with user "user1":

```
# htrealm add -i sws_server -h www.A.com -r Project \\
-s HTPASSWD -d realms/Project

# htrealm add -i sws_server -h www.A.com -r Project -u user1
Setting password for the user user1.
Password:
Confirm Password:

# htaccess add -i sws_server -h www.A.com -U "/project" \\
 -r Project -s BASIC -m GET -u '*' -y
```

CODE EXAMPLE 1

To delegate access control management to the user webmaster in the realm `WebUsers`:

```
# htaccess add -i sws_server -h www.A.com -U / -r WebUsers -s MD5 -a \\
 -u webmaster
```

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/access.conf` A web site's access control lists configuration file.

`/etc/http/access.conf`	The server administration access control lists configuration file.
site_path`/conf/realms.conf`	Defines the realms used to define users for web site access control lists.
`/etc/http/realms.conf`	Defines the realms used to define users for server administration.

SEE ALSO **`access.conf`**(4), **`htrealm`**(1m), **`realms.conf`**(4)

NOTES If the command is run by `root` user, then the username and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

htcontent – Administers HTTP/1.1 meta-data associated with resources in a document tree.

SYNOPSIS

htcontent add –h *hostname* –i *instance* –n *uri* [–p] [–u *administrator*][–O *directory_options* | –P *preference_options* | –V *variant_options*]

htcontent delete –h *hostname* –i *instance* –n *uri* [–p] [–u *administrator*][–O *directory_options* | –P *preference_options* | –V *variant_options*]

htcontent help

htcontent list –h *hostname* –i *instance* –n *uri* [–p] –u *administrator*[–O *directory_options* | –P *preference_options* | –V *variant_options*]

htcontent version

DESCRIPTION

`htcontent` creates or deletes information about the content of resources on a web site.

For directories, you can set whether they can be browsed and also sets the format for directory listings. You can also create a list of default file names to search for in the directory.

For files, you can set preferences for HTTP 1.1 content negotiation. Preferences include character set, language, compression encoding, and media type.

You can set variant information for a Uniform Resource Identifier (URI). If a URI has a set of associated file variants, the server will select the most appropriate variant based on the client's preferences and the preference settings on each file variant.

See the Examples section for more detail.

OPTIONS

Subcommands

The following subcommands are supported:

add	Adds directories, options, preferences, or variants for a URI.
delete	Deletes directories, options, preferences, or variants for a given URI.
help	Displays help on usage.
list	Lists the configured directories, options, preferences, or variants for a URI.

version Displays the version of htcontent.

Options

The following options are supported:

–h ***hostname*** Specifies the virtual host name. Required with all subcommands.

–i ***instance*** Specifies the name of the httpd instance. Required with all subcommands.

–n ***uri*** Specifies the URI. Required with the all subcommands.

–O ***directory_options*** Specifies options for directory listings and default files on directory options. Valid with all subcommands. This option requires a comma-separated list of parameters to specify directory settings. Lists should be specified in order of preference. The following are the valid directory options:

a ***all*** Indicates delete all preference or variants information. Valid with the delete subcommand.

d **[=*listing_type*]** Sets the method for displaying the contents of a directory when there is no file matching one of the default file names. Valid with the add and delete subcommands. The *listing_type* directive can be one of the following:

`fancy` Displays each directory as a hyperlink with the file size, the last modified

		time, and an icon next to each entry to indicate the file type.
	`off`	Displays no directory contents.
	`simple`	Displays directory entries as plain text hyperlinks.
`f` **[=** ***file*****]**	Specifies the file associated with the variant information (–V) or a list of default file names in a directory (–0). Use a colon (:) to separate items in a list. File names must be relative to the URI. Valid with the add and delete subcommands.	

–P ***preference_options*** Sets the server's content negotiation preferences for the specified URI. Valid with all subcommands. This option requires a comma-separated list of parameters to specify preference settings. Lists should be specified in order of preference. The following are the valid preference options:

a ***all***	Indicates delete all preference or variants information. Valid with the delete subcommand.
c **[=** ***charset*****]**	Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters. The default *charset* for variants is ISO-8859–1. Valid with the add and delete subcommands.
e **[=** ***encoding*****]**	Specifies the preferred encodings or the encoding type of a variant. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals which methods have been used to encode the file. For preferences, separate multiple encodings with a colon (:). Valid with the add and delete subcommands.
l **[=** ***lang*****]**	Specifies the preferred languages of a variant. Languages are specified in the standard two-letter format. For preferences, separate multiple languages with a colon (:). Valid with the add and delete subcommands.
t **[=** ***media_type*****]**	Specifies the preferred types of media of a variant. Media type is in standard MIME type format. For preferences,

separate multiple media types with a colon (:). Valid with the add and delete subcommands.

–p Disables password prompting. Passwords will be read from stdin. Valid with the add and delete subcommands.

–u ***administrator*** Specifies a user name in the web site's administration realm (`serverAdmin` by default), or in the realm specified by the ACL from the URI. Required with all subcommands.

–V ***variant_options*** Specifies variant suboptions. Valid with all subcommands. This option requires a comma-separated list of parameters to specify variant settings. Lists should be specified in order of preference. The following are the valid variant options:

a ***all***	Indicates delete all preference or variants information. Valid with the delete subcommand.
c **[=** ***charset*****]**	Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters. The default *charset* for variants is ISO-8859–1. Valid with the add subcommand.
e **[=** ***encoding*****]**	Specifies the preferred encodings or the encoding type of a variant. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals which methods have been used to encode the file. For preferences, separate multiple encodings with a colon (:). Valid with the add subcommand.
f **[=** ***file*****]**	Specifies the file associated with the variant information (–V) or a list of default file

	names in a directory (–0). Use a colon (:) to separate items in a list. File names must be relative to the URI. Valid with the add and delete subcommands.
`l` **[=** ***lang*****]**	Specifies the preferred languages of a variant. Languages are specified in the standard two-letter format. For preferences, separate multiple languages with a colon (:). Valid with the add subcommand.
`t` **[=** ***media_type*****]**	Specifies the preferred types of media of a variant. Media type is in standard MIME type format. For preferences, separate multiple media types with a colon (:). Valid with the add subcommand.

EXAMPLES

EXAMPLE 1

This example displays a page named `home.html` in English, French, or German based on the client's preference. There are three files: `home.en.html`, `home.fr.html`, and `home.de.html`.

```
# htcontent add -i sws_server -h www.A.com -n home.html \\
   -u admin -V f=home.en.html,l=en
 Enter Password for admin:

# htcontent add -i sws_server -h www.A.com -n home.html \\
   -u admin -V f=home.fr.html,l=fr
 Enter Password for admin:

# htcontent add -i sws_server -h www.A.com -n home.html \\
   -u admin -V f=home.de.html,l=de
 Enter Password for admin:
```

CODE EXAMPLE 1

To view the variants associated with a URI, use `htcontent list`. After Example 1, you could verify the content settings:

```
# htcontent list -i sws_server -h www.A.com -n home.html \\
  -u admin -V
Enter Password for admin:
  home.de.html   lang = de
  home.fr.html   lang = fr
  home.en.html   lang = en
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/content.conf` Defines the content variants, encoding types, and directory preferences for a web site.

SEE ALSO

`content.conf`(4)

NOTES

If the command is run by `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

hthost – Create, delete, and manage web sites on Sun™ WebServer™ `httpd` instances.

SYNOPSIS

hthost add [–f *site_config_file*] [–g *groupname*] –h *hostname* –i *instance* [–p] –s *site_path* [–u *username*] –z *admin*

hthost delete –i *instance* –h *hostname* [–p] –z *admin*

hthost disable –i *instance* –h *hostname* [–p] –z *admin*

hthost enable –i *instance* –h *hostname* [–p] –z *admin*

hthost help

hthost list [–i *instance*–h *hostname*] [–p] –z *admin*

hthost version

DESCRIPTION

`hthost` is used primarily to add and delete web sites. The `add` subcommand associates a web site (or virtual host) with system resources, such as the server instance that hosts the site, a configuration directory, a configuration file, and a host name.

The `enable` and `disable` subcommands are used to activate or shut down web sites on a running server instance. The enabled or disabled state is saved so that if the server instance is restarted, enabled sites are automatically reactivated.

Once a web site has been created on a server instance, you must edit its configuration file to further customize the site; see **`httpd.site.conf`**(4).

OPTIONS

Subcommands

The first argument to `hthost` must be one of the following subcommands:

add Adds a web site configuration for a host name to a server instance. The `add` subcommand creates a site path for configuration directories, creates a configuration file, and grants administrative rights to the given user name and optional group. The initial state of the site is enabled.

delete Deletes a web site configuration from a server instance. References to the site in the configuration files for the server instance are deleted, and the site is no longer available through `httpd`. The site's directories and files remain in place.

disable Disables a web site. The server instance that serves the site will not respond to requests for disabled sites.

enable Enables a web site, making it available through the server instance.

help Displays usage information for the command.

list Lists the sites supported by a server or properties of a specific site. If only an instance name is supplied, a list of all web sites on that server displays. If an instance name and a hostname are supplied, then details about the web site for the host name are displayed.

version Displays the version of the `hthost` command.

Subcommand Options

The following options are supported.

–`f` ***site_config_file*** Specifies the location of the site configuration file relative to the *site_path* specified by –`s`. By default, site configuration is stored in the *site_path*`/conf` directory. Valid only with the `add` subcommand.

[–`g` ***group_name*****]** Specifies a group in the server administration realm that has ownership rights on the new site. Valid only with the `add` subcommand.

–`h` ***hostname*** Specifies the host name of the site to which a subcommand applies. The host name is a token used to identify the site; no name service lookups are performed, for example, to expand a host name to a fully qualified domain name. Any form of the host name may be used for the `add` subcommand, but other commands must use the same form used when the site was added. Valid with all subcommands.

–`i` ***instance*** Specifies the server instance which hosts the web site. The *instance* name is defined uniquely for each server when it is created. Valid with all subcommands.

–`s` ***site_path*** Specifies the absolute path to the web site's directory tree. The *site_path* contains all configuration, access control, realm, and content

directories and files for the site. Valid only with the `add` subcommand.

`-u` ***username*** — Specifies a user in the server administration realm that has ownership rights on the new site. Valid only with the `add` subcommand.

EXAMPLES

EXAMPLE 1

A server administrator named `serverAdmin1` creates a site named www.A.com. The site will have an administrator named `user1` and have all configuration files and public documents in `/opt/WWW/A.com/`.

In order for `hthost` to successfully create a directory for the new site, you must have write permission to the directory under which the site configuration files will reside.

```
# hthost add -i sws_server -h www.A.com -u user1 \\
  -s /opt/WWW/A.com -f conf/A.com.httpd.conf -z serverAdmin1
Enter Password for serverAdmin1:

Creating site directory:
  /opt/WWW/A.com
Creating site configuration:
  /opt/WWW/A.com/conf/A.com.httpd.conf

# ls /opt/WWW/A.com
cgi-bin/        public/
conf/           servlets/
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/`*site_name*`.site.conf` The web site configuration file.

`/etc/http/`*instance_name*`.httpd.conf` The server instance configuration file. When `hthost` adds a new site, it creates an entry in `httpd.conf` to defined the *site_path* and web site configuration file.

SEE ALSO **`httpd.site.conf`**(4), **`httpd.conf`**(4)

NOTES If the command is run by `root` user, then the username and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

htmap – Manages maps at the site level.

SYNOPSIS

htmap add [–c *class*] –i *instance* –f *from* –h *hostname* –t *to* [–z *admin*[–p]]

htmap delete –i *instance* –f *from* –h *hostname* [–z *admin*[–p]]

htmap help

htmap list –i *instance* [–f *from*] –h *hostname* [–z *admin*[–p]]

htmap version

DESCRIPTION

The htmap command adds, deletes, and lists aliases from one Uniform Resource Identifier (URI) to another resource on a web site. By default it creates a reference from a URI token to a file or directory on disk.

htmap administers maps to establish an alias to another resource, make a resource outside of the doc_root accessible to a client, or partition the name space into various classes of resources such as CGI, imagemap, or servlet.

OPTIONS

The following subcommands are supported:

add	Adds a new map.
delete	Deletes an existing map.
help	Displays help on usage.
list	Lists all maps.
version	Displays the version of htmap command.

The following options are supported:

–c ***class*** Specifies the class file for the map. Values can be one of the following (if no –c is specified, then the class defaults to NULL):

ADMIN	Treats the alias as a URL to access the Sun WebServer Administration Console and the administration server.
CGI	Treats the aliased file or directory as a CGI resource (all files located here will be treated as executable scripts).

IMAP	Treats the alias as an imagemap resource.
NULL	Treats the aliased directory in no special way.
REMOTE	Treats the alias as a new URL, either on the local host or on another network location.
SERVLET	Treats the aliased *resource_target* as a servlet or a chain of servlets.
STATS	Treats the alias as an interface to server statistics.

–f ***from***	Indicates the URI token the web server will map. Any URI that begins with this token will be redirected to the resource defined by the map class and the –t to destination.
–h ***hostname***	Specifies the virtual host.
–i ***instance***	Specifies the name of the httpd instance. Valid with all subcommands.
–t ***to***	Defines the path name or URL to the actual resource. Valid with the add subcommand.

EXAMPLES

EXAMPLE 1

To create a URL `http://www.A.com/swshelp/` that references a directory outside of the `www.A.com` document root:

```
# htmap add -i sws_server -h www.A.com -f /swshelp/ \\
   -t /usr/http/admin_server/public/admin/help/en/
```

CODE EXAMPLE 1

To create a URL `http://www.A.com/siteadmin/` that starts the Sun WebServer Administration Console for administration of the web site:

```
# htmap add -i sws_server -h www.A.com -f /siteadmin/ \\
  -t /sws-administration -c ADMIN
```

CODE EXAMPLE 2

This example shows how to create an alias that accesses a servlet without using the standard servlet token (`/servlet/` by default). To redirect `http://www.A.com/calendar/` to a servlet chain that invokes a servlet named `login` then a servlet named `calendar`:

```
# htmap -i sws_server -h www.A.com -f /calendar/ \\
  -t login,calendar -c SERVLET
```

Note that `login` and `calendar` must be in the servlets path and have definitions in `servlets.properties`.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/map.conf` Creates an alias to a path on the file system or a redirection to a remote URL from a URI on the host..

SEE ALSO

`map.conf`(4)

NOTES

If the command is run by `root` user, then the username and password of an administrator are not required.

Users other than `root` must use the –z option and pass the user name and password of a valid administrator to the command.

NAME

htpasswd – Administers passwords for the users in HTPASSWD realms.

SYNOPSIS

htpasswd help

htpasswd version

htpasswd [–i *instance_name*] [–h *hostname*] [–p] –r *realm_name* –u *user_name* [–z *admin_name*]

DESCRIPTION

`htpasswd` is a utility used only to change passwords for users in HTPASSWD realms. It is provided as a tool that can be incorporated in CGI or other scripts to automate password maintenance.

Users must be created using **`htrealm`**(1m). Once users are created, any system user can run `htpasswd` to update passwords (as long as the realm administrator name and password are specified).

OPTIONS

The following options are supported:

–i ***instance***	Specifies the name of the server instance. Valid with all subcommands.
–h ***hostname***	Specifies the name of the virtual host containing the realm. Valid with all subcommands.
–p	Turns off password prompting (for scripts). Valid with all subcommands.
–r ***realm_name***	Specifies the realm name. Valid with all subcommands.
–u ***user_name***	Specifies a user name whose password is to be set so that the user can have permission to modify realm data. Separate multiple user names with a white space. Valid with all subcommands.
–v	Displays verbose status messages.
–z ***admin_name***	Specifies the name of the administrator of the server, web site, or realm. If users omit this option, users will be prompted for the current password and then the new password. This allows users to change the passwords themselves. Valid with all subcommands.

EXAMPLES

EXAMPLE 1

Any user can change passwords in an `HTPASSWD` realm if they have the user name and password of the realm administrator. If `realmadmin` is the realm administrator name, a user (or CGI script) can change the password for `user1`:

```
% htpasswd -i sws_server -h www.A.com -r Project \\
  -u user1 -z realmadmin
 Enter Password for realmadmin:
 Setting pasword for the user user1.
 Password:
 Confirm password:
```

CODE EXAMPLE 1

The user `user1` in the `WebUsers` realm changes his own password:

```
 % htpasswd -h www.A.com -i sws_server -r WebUsers -u user1
                          Password for user user1: ***
                          Changing password for the user user1
                          New Password: *****
                          Confirm Password: *****
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/realms.conf` Defines realms of user and group information used by access control lists on a Sun WebServer web site.

SEE ALSO **realms.conf**(4), **htrealm**(1m)

NOTES

In order to access the global HTPASSWD realms (`/etc/http/realms/conf`), omit the –`i` and –`h` flags.

This command is installed with setuid to `adm` to permit end users invoking the command to have write access to the Sun WebServer configuration after performing necessary checks.

Superusers do not need to specify the –`z` flag, and are allowed to access any command-line utility without authentication.

If the command is run by `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the –`z` option and pass the user name and password of a valid administrator to the command.

The server will check whether a user has been designated the realm administrator, site administrator, or server administrator (in this order), where the later two are defined as those principans who have access to the pseudo-URI ("`/sws-administration`")at the site-level and global access control configuration (`/etc/http/access.conf`).

A server administrator has access to site administration and is able to manage site realms, ACLs, and content. However, site administrators can override this setting by delegating administrators in the administrator blocks in realms, ACLs, and content configuration.

NAME

htrealm – Manages realms, users, and groups used to configure Access Control Lists on a Sun™ WebServer™ web site.

SYNOPSIS

htrealm add [–A] [–d *data_dir*] [–g *groupname*] [–h *hostname*–i *instance*] [–m *individuals*] [–p] –r *realmname* –s[[UNIXSYS]|[HTPASSWD]] [–u *userid*] –v [–z *admin_name*]

htrealm delete[[–A]|[–m *individuals*]] [–g *groupname*] [–h *hostname*–i *instance*] [–p] –r *realmname* –u *userid* –v [–z *admin_name*]

htrealm help

htrealm list[[–A]|[–u *userid*]|–g *groupname*|–r *realmname*] [–h *hostname*–i *instance*] [–p] –v [–z *admin_name*]

htrealm version

DESCRIPTION

Realms in Sun WebServer define sets of protection spaces or authentication domains consisting of user names, groups, and passwords. Access controls based on password authentication use a realm to determine how to authenticate users. For example, a person may have a user name and password for a UNIX® system account and a different set of credentials for Sun WebServer administration. The realm used to protect a resource on a web site would determine which password the person would need.

Realms are also differentiated based on how they are used. You can define two realms, for example, that both use UNIX user names and passwords. One realm could be named “admins” and another “subscribers.” Both use the same set of valid UNIX system users, but in practice you might only use “admins” with access control lists (ACLs) that are restricted to a small set of users. When prompted to enter a password for the “admins” realm or the “subscribers” realm, the user knows which password to enter and whether the user has access to the data (that is, the user knows the purpose for authenticating). If “admins” and “subscribers” were HTPASSWD realms, they will have different ACLS and user names.

Most browsers display the realm name in the prompt when a user name and password are required, so the realm name should indicate to users the purpose for password protection and which user name and password to use.

The htrealm command can be used to create, delete, and list realm definitions for use with ACLs. It can also be used to manage users and groups in HTPASSWD realms. HTPASSWD realms are Sun WebServer specific in that their data is stored in user and group files with Sun WebServer configuration.

Realms in the global `/etc/http/` directory are independent of any web site. These realm definitions are used only for server administration; the user names and passwords are used to log into the Sun WebServer Administration Console or to execute commands such as `htserver`. Only one such realm may be in use at any given time. The server administration realm must be defined in `/etc/http/realms.conf` and used to protect the `/sws-administration` URI in `/etc/http/access.conf`.

OPTIONS

Subcommands

The following subcommands are supported:

add	Adds a given realm, user, group, or memeber.
delete	Deletes a given realm, user, group, or member.
help	Displays help on usage.
list	Lists all realms, users, groups, or memebers.
version	Displays the version of `htrealm`.

Options

The following options are supported:

–A	Indicates that the user or group specified with the –u or –g flags has administrative privileges of the ACL. The administrators must already be valid principals within the realm. Valid with all subcommands (but used most frequently to add, delete, or list realm administrators).
–d ***data_dir***	Specifies a directory relative to the site path where the `users` and `groups` files for an `HTPASSWD` realm are stored. `data_dir` is required and valid only if –s is `HTPASSWD`, or if you're running on Solaris for ISPs, `ISPADMIN`. The default is *site_path*/conf/realms/*realmname*. Valid with the add subcommand.
–g ***groupname***	Specifies a set of users with permission to access the resources in the realm. Separate multiple group names with white space. Valid with all subcommands.
–h ***hostname***	Specifies the name of the virtual host containing the realm. Valid with all subcommands.
–i ***instance***	Specifies the name of the server instance. Valid with all subcommands.

–m ***individuals***	Specifies the individual members of the group. This is a comma-separated list. Valid with the add and delete subcommands.
–p	Turns off the prompting for the password such that passwords are taken in from stdin and scripts may pipe (|) passwords. Valid with all subcommands.
–r ***realmname***	Specifies the realm name. White spaces must be inside double quotes. Valid with all subcommands.
–s ***source_name***	Specifies the source of the realm (UNIXSYS or HTPASSWD). Valid with the add subcommand.

HTPASSWD	Indicates that the user or group information is retrieved using the Sun WebServer users/group file format, and that user and group information will be maintained in the data directory named by `realm_dir`. The **`htrealm`**(1m) utility is used to create, delete, and list users and groups and modify passwords using `htpasswd`.
ISP	Indicates that the realm information is stored in the Solaris™ for ISPs shared directory service. Changes to user and group information cannot be made through Sun WebServer.
ISPADMIN	Indicates that the principals are Administrators in the Solaris™ for ISPs™ Management Console. The –d flag takes the ISP-component ID and version (for example, “SUNWfinger–1.0”).
UNIXSYS	Indicates that the operating system user and group definitions will be used to

authenticate users in the realm. Changes to user and group information cannot be made through Sun WebServer.

–u ***userid*** Specifies the realm user with permission to modify realm data. Separate multiple user names with white space. Valid with all subcommands.

–v Displays verbose status messages. Valid with all subcommands.

–z ***admin_name*** Specifies the name of the realm administrator. Valid with all subcommands.

EXAMPLES

EXAMPLE 1

To create a site-specific realm called `Subscribers` on the web site `www.A.com`, you create at least one user and one realm administrator:

```
# htrealm add -i sws_server -h www.A.com -r Subscribers \\
  -s HTPASSWD

# htrealm add -i sws_server -h www.A.com -r Subscribers \\
  -u user1
Setting password for the user user1.
Password:
Confirm Password:

# htrealm add -i sws_server -h www.A.com -r Subscribers \\
  -u user1 -A
```

CODE EXAMPLE 1

A nonroot user can add a realm if a valid user name and password from the `serverAdmin` realm are supplied:

```
% htrealm add -i sws_server -h www.A.com -r System \\
  -s UNIXSYS -z admin
Enter Password for admin:

% htrealm list -i sws_server -h www.A.com -z admin
Enter Password for admin:
```

(continued)

(Continuation)

siteAdmin HTPASSWD -
System UNIXSYS -

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/access.conf` A web site's ACLs configuration file.

`/etc/http/access.conf` The server administration ACLs configuration file.

site_path`/conf/access.conf` A web site's ACLs configuration file.

`/etc/http/access.conf` Defines the realms used to define users for server administration.

site_path`/conf/realms.conf` Defines realms of user and group information used by access control lists on a Sun WebServer web site.

site_path`/conf/realms/`*HTPASSWD_realm_name*`/HTPASSWD` Lists the users in the HTPASSWD realm.

Entries in this file have the form *username*:*password*.

site_path`/conf/realms/`*HTPASSWD_realm_name*`/GROUPS` Lists the groups in the HTPASSWD realm.

Entries in this file have the following form:

```
group <group_name> {
 member1
 member2
 member3
}
```

SEE ALSO **realms.conf**(4), **htaccess**(1m), **access.conf**(4), **htpasswd**(1m)

NOTES If the command is run by `root` user, then the username and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

htserver – Create, delete, and manage Sun™ WebServer™ `httpd` instances.

SYNOPSIS

htserver add *instance* [*conf_file*]

htserver delete *instance* [*instance*...]

htserver disable [[–a][*instance*]...]

htserver enable [[–a][*instance*]...]

htserver help *command*

htserver list [[–a][*instance*]...]

htserver query [[–a][*instance*]...]

htserver restart [[–a][*instance*]...]

htserver start [[–a][*instance*]...]

htserver stop [[–a][*instance*]...]

htserver update *instance* [*conf_file*]

DESCRIPTION

The `htserver` command creates and maintains Sun WebServer server instances. Each server instance is a process associated with a configuration file, and each one hosts one or more web sites.

`htserver` can start, stop, and restart server instances. It can also enable or disable server instances. Each “enabled” server instance will be started when the machine reboots or when `htserver start` is run with no instance specified.

The `list` and `query` subcommands can be used to get information about what servers are running or enabled and what configuration files each instance uses.

Once a server instance is created, use **`hthost`**(1m) to add web sites. To modify the configuration, either use the Sun WebServer Administration Console (http://host:2380/admin/admin.html), or edit the server configuration file (see **`httpd.conf`**(4)).

The server instance named `admin` is the administration server. This server instance is created when Sun WebServer is installed and listens to port 2380 on all IP addresses. The `admin` server instance is used to access the Sun WebServer Administration Console.

OPTIONS

Subcommands

The first argument to `htserver` must be one of the following subcommands:

add Creates a reference to a new server instance so that it can be managed. An entry is added to the server list in **`httpd-instances.conf`**(4).

delete Deletes a server instance from httpd-instances.conf. This removes the server instance from Sun WebServer management, but does not delete the data or configuration files for the server or any of its sites.

disable Disables a server instance. Disabled instances can only be started by running `htserver start` *instance_name*, and explicitly specifying the instance name. Disable does not stop a running server instance.

enable Enables a server instance. Enabled instances will be started when the machine reboots or whenever `htserver start` is run with no instance named.

help Displays usage information for this command.

list Lists server instances and status information for all servers or each named instance.

query Displays detailed status and statistical information about a named server instance.

restart Restarts named server instances or all of the currently running server instances.

start Starts named server instances or all enabled server instances with the configuration files listed in httpd-instances.conf.

stop Stops named server instances or all running server instances.

update Updates a named instance with the named configuration file.

Subcommand Options

The following options are supported:

–a Indicates that the command applies to the Sun WebServer Administration server. The Sun WebServer Administration server is a special server instance that allows remote administration of servers and sites through the Sun WebServer Administration Console.

–v Runs the command in verbose mode with more descriptive messages output to the screen.

OPERANDS

The following operands are supported:

conf_file Specifies the name of a configuration file to use for a server instance. This operand is required as the last argument for the `add` and `update` subcommands.

instance Specifies a server by its instance name. Instance names are maintained in the `/etc/http/httpd-instances.conf` file. A single instance name is required with the `add`, `update`, `delete`, and `query` subcommands. Other subcommands take an optional instance name or list of instance names.

EXAMPLES

EXAMPLE 1

To list all server instances (nonroot users must use `---z` and provide a user name and password from the `serverAdmin` realm):

```
# htserver list -z admin
Enter Password for admin:

Instance : sws_server
        Enabled : No
        pid : -
        Config file : / /etc/http/sws_server.httpd.conf

Instance : aws
        Enabled : Yes
        pid : 4018
        Config file :  /var/opt/SUNWixamc/awsconf/aws.conf

Instance : SUNWixmon
        Enabled : Yes
        pid : 4020
        Config file : /opt/SUNWixmon/sws/SUNWixmon.httpd.conf

Instance : admin
        Enabled : No
        pid : -
        Config file : / /usr/http/admin_server/conf/admin.httpd.conf
```

To create a new server named `Large_Sites` with default values for the configuration file (`/etc/http/Large_Sites.httpd.conf`), server root directory (`/var/http/Large_Sites`), and site directory (`/var/http/Large_Sites/websites/`):

```
# htserver add Large_Sites
Creating server configuration file:
 /etc/http/Large_Sites.httpd.conf
Creating server root directory:
/var/http/Large_Sites
Creating site directory:
/var/http/Large_Sites/websites/default_site
Creating site configuration:
/var/http/test_server/Large_Sites/websites/default_site/conf/default_site.site.conf
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

`/etc/http/`*instance_name*`.httpd.conf` The server instance configuration file.

`/etc/http/httpd-instances.conf` Tracks all Sun WebServer instances. When `htserver` creates a new instance, an entry is added to this file.

SEE ALSO

`httpd.conf`(4), **`httpd-instances.conf`**(4), **`hthost`**(1m)

NOTES

If the command is run by `root` user, then the username and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

htservlet – Configures the behavior of a servlet engine.

SYNOPSIS

htservlet add [–a *init_args*] [–b *codebase*] –c *classname* –h *hostname* –i *instance* –n *servlet_name* [–p] [–s] –u *username* [–v]

htservlet delete –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet disable –g *option* [–h *hostname*] –i *instance* [–p] –u *username* [–v]

htservlet enable –g *option* [–h *hostname*] –i *instance* [–p] –u *username* [–v]

htservlet help

htservlet list –h *hostname* –i *instance* [–v]

htservlet load –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet log [–f *prefix*] [–h *hostname*] –i *instance* [–m *max_num_files*] [–p] [–t *cycle_time*] –u *username* [–v] [–z *file_size*]

htservlet reload [–a *init_args*] –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet query –h *hostname* –i *instance* [–v]

htservlet security [–h *hostname*] –i *instance* [–o *permission*] [–p] [–r *resource*] –u *username* [–v]

htservlet settings [–d *servlet_path*] [–h *hostname*] –i *instance* [–j *server_classpath*] [–P *properties_file*] [–p] –u *username* [–v]

htservlet unload –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet version

DESCRIPTION

A servlet engine can be defined on any Sun WebServer server instance or any individual web site. The servlet engine runs a Java™ virtual machine that loads and executes servlets defined in its `servlets.properties` file.

The `htservlet` command defines and modifies servlet engine run time, security, and logging properties; adds or removes entries in the `servlets.properties` file; and loads or unloads servlets in running servlet engines.

When –h *hostname* is omitted, the subcommand applies to the shared servlet engine (server-wide setting).

OPTIONS

Subcommands

The following subcommands are supported:

Note - You must restart the server in order for the changes made to the configuration files to take effect.

---

add	Adds a servlet to the `servlets.properties` file. Adding a servlet does not imply that the servlet is automatically loaded.
delete	Deletes a servlet from the servlet engine and writes changes to the `servlets.properties` file.
disable	Disables an option on the servlet engine and writes changes to the configuration file.
enable	Enables an option on the servlet engine and writes changes to the configuration file.
help	Displays help on usage.
list	Lists all loaded servlets on the server.
load	Loads a servlet from the `servlets.properties` file on a running server.
log	Configures the log location and cycling parameters and writes changes to the configuration file.
query	Returns current servlet engine settings on a running server.
reload	Reloads a servlet on a running server.
security	Configures the security settings for the servlet engine and writes changes to the configuration file.
settings	Specifies the settings on the servlet engine and writes changes to the configuration file.
unload	Unloads a servlet from the running web server.
version	Displays the version of htservlet.

Options

The following options are supported:

–a ***init_args*** Specifies the optional initial arguments passed to the servlet. Used in the format *name=value* [, *name=value*...]. Valid with the add and reload subcommands.

–b ***codebase*** Specifies the URL of the servlet's code base. This URL can be pointing to a directory or a jar file. Used only for remote servlets. Valid with the add subcommand.

–c ***classname*** Specifies the name of the servlet main class file. Valid with the add subcommand.

–d ***servlet_path*** Specifies the directories and jar files for the servlet engine on the local machine. This is a colon-separated list. This option can only prepend the path specified to the original path in the file. Valid with the settings subcommand.

–f ***prefix*** The path name and the file prefix for servlet log files. As new log files are created, they use this prefix and a number suffix. Valid with the log subcommand.

–g ***option*** Specifies a servlet engine option to enable or disable. Valid options are:

`chain`	Enables or disables servlet chaining. Servlet chains are a sequence of servlets executed in the specified order to fulfill one single servlet request.
`jvm`	Enables or disables the server to enable a jvm. This is a server-wide setting.
`log`	Enables or disables servlet error logging.
`remote`	Enables or disables loading servlets from remote sites in this servlet engine.
`se`	Enables or disables the servlet engine for server process or web site.

`share`	Allows all sites on the server to share a single servlet engine. This is a server-wide setting. If this option is disabled, each site is allowed to create its own servlet engine instance.

`–h` ***hostname*** Specifies the name of the virtual host. Valid with all subcommands.

`–i` ***instance*** The name of the server instance. Valid with all subcommands.

`–j` ***server_classpath*** Specifies the class path for the jvm which may include the location of `classes.zip` file of JDK, JSDK, and the servlet engine. This is a colon separated list (for example, *usr*/*lib*/*java*/ [:/*usr*/*java*/*lib*...]). This command only prepends to the existing path. It is a server-level setting. Valid with the settings subcommand.

Note - Do not put the servlets directories in the server classpath.

`–m` ***max_num_files*** Specifies the maximum number of log files. When the log suffix exceeds this number, the next log file is created overwrites the first log file. The default number is 7 files. Valid with the log subcommand.

`–n` ***servlet_name*** Specifies the name of the servlet. Valid with the add, delete, load, reload, and unload subcommands.

`–o` ***permission*** Specifies the access settings used in conjunction with the `–r` option. Valid with the security subcommand.

`all`	Allows local and remote servlets to access resources.
`local`	Allows access only to resources on the same host.
`none`	Allows no access.
`remote`	Allows access only to resources on servlets with a code base.

`–P` ***properties_file*** Specifies the full path to the properties file for the servlet engine. Valid with the settings subcommand.

`–p` Disables password prompting, and password is piped (“|”) to the command. Valid with the add, delete, disable, enable, load, log, reload, unload, security, and settings subcommands.

`–r` ***resource*** Specifies the resource settings used in conjunction with the `–o` option for access control. Valid with the security subcommand.

`file`	Sets access permissions for file resources such as read / write a file on local disk.
`link`	Sets access permissions for links to dynamic libraries.
`network`	Sets access permissions for network resources.
`security`	Sets access permissions for security resources such as `classLoaders`.
`system`	Sets access permissions for system resources such as **System.Exec ()**.

`–s` Indicates that the servlet will be loaded at startup. Valid with the add subcommand.

`–t` ***cycle_time*** The log cycle time measured in minutes. When the log cycle time exceeds this number, a new log file is created with an incremented suffix. The default time is 1440 minutes (1 day). Valid with the log subcommand.

`–u` ***username*** Specifies user name. Valid with the add, delete, disable, enable, load, log, reload, unload, security, and settings subcommands.

`–v` Specifies verbose mode for more detailed messages. Valid with all subcommands.

–z ***file_size*** Specifies the maximum log file size measured in bytes. When the log file size exceeds this number of bytes, a new log file is created with an incremented suffix. The default file size is 1048576 bytes (1 MB). Valid with the log subcommand.

EXAMPLES

EXAMPLE 1

To enable servlets on a server:

```
# htservlet enable -g jvm -i sws_server -u admin
```

EXAMPLE 2

To load a declared servlets on a server:

```
# htservlet load -i sws_server -h www.A.com -n foo -u http
```

EXAMPLE 3

To add servlets to be loaded at startup:

```
# htservlet add -i sws_server -h www.A.com -n foo \\
-c FooServlet -b http://x.eng/ -a counter=1 -s \\
-u http
```

EXAMPLE 4

To add a servlet declaration (in verbose mode):

```
# htservlet add -i sws_server -h www.A.com -n foo \\
-c FooServlet -b http://x.eng/ -a counter=1 -v \\
-u http
```

EXAMPLE 5

To set the server classpath:

```
# htservlet settings -i sws_server -j /usr/jdk/lib/classes/zip:. \\
-u http
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtsvl
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/`*site_name*`.httpd.conf` Contains the web site servlet engine configuration if the servlet engine is not shared.

`/etc/http/httpd.conf` Contains the server instance servlet engine configuration if all web sites share the servlet engine.

site_path`/conf/servlets.properties` Defines each servlet that can be loaded by a web site.

SEE ALSO

`httpd.conf`(4), **`httpd.site.conf`**(4), **`servlets.properties`**(4)

NOTES

If the command is run by `root` user, then the username and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

`/etc/init.d/httpd` – Starts and stops servers.

SYNOPSIS

httpd help

httpd start

httpd stop

DESCRIPTION

Server instances can be started or stopped by using the `htserver` utility, through the Sun WebServer Administration Console, or by executing this script. It is recommended that you use `htserver` or the Sun WebServer Administration Console.

OPTIONS

The following subcommands are supported:

help	Displays help on usage.
start	Starts all “enabled” servers in `httpd-instances.conf`.
stop	Stops all servers.

–

–

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

`htserver`(1m)

NOTES

man Pages(4): File Formats

NAME

httpd.intro – Introduction to the man pages of the Sun™ WebServer™ configuration and log files. The man pages offer detailed instruction and examples on syntax and directives for each file.

DESCRIPTION

The configuration and log files are available to configure and monitor Sun WebServer.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtman
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/access.conf` A web site's Access Control Lists (ACLs) configuration file.

`/etc/http/access.conf` The server administration ACLs configuration file.

site_path`/conf/access.conf` A web site's ACLs configuration file.

`/etc/http/access.conf` Defines the realms used to define users and groups for server administration.

site_path`/conf/content.conf` Defines the content variants, encoding types, and directory preferences for a web site.

`/etc/http/httpd-instances.conf` Tracks all Sun WebServer instances. When `htserver` creates a new server instance, an entry is added to this file.

site_path`/conf/`*site_name*`.httpd.conf` Contains the web site servlet engine configuration if the servlet engine is not shared.

`/etc/http/`*instance_name*`.httpd.conf` The server instance configuration file. When `hthost` adds a new site, it creates an entry in `httpd.conf` to define the *site_path* and web site configuration file.

site_path`/conf/`*site_name*`.site.conf` The web site configuration file.

`/etc/http/`*instance_name*`httpd.conf` Contains the server instance servlet engine configuration if all web sites share the servlet engine.

site_path/conf/map.conf	Creates an alias to a path on the file system or a redirection to a remote URL from a Uniform Resource Identifier (URI) on the host.
site_path/conf/realms.conf	Defines realms of user and group information used by access control lists on a Sun WebServer web site.
site_path/conf/servlets.properties	Defines each servlet that can be loaded by a web site.

SEE ALSO

htaccess(1m), **htcontent**(1m), **hthost**(1m), **htmap**(1m), **htpasswd**(1m), **htrealm**(1m), **htserver**(1m), **htservlet**(1m), **httpd**(1m)

NOTES

access.conf	Defines ACLs for the content hosted by a web site. ACLs regulate access to resources on the site by defining which users, groups, and/or hosts have permissions to make HTTP GET, PUT, POST, and DELETE requests.
content.conf	Represents meta-data for directories, site preferences for languages, media and encoding types, and specifying variants for content.
httpd-instances.conf	Associates each unique `httpd` process (or server instance) with its configuration file, and defines whether or not a server instance is enabled.
httpd.cgi.logs	Logs errors generated by CGI scripts. Log files will be named *prefix.sequence*, where *sequence* is a cycling number.
httpd.conf	The server instance configuration file contains directives that define the server's runtime behavior, the web sites it hosts, and the network connection end points it uses.
httpd.event.logs	The server events log files. The `httpd` server puts out error messages and warnings via `syslogd` to `/var/adm/messages` by default.
httpd.request.logs	Logs all incoming requests to a server activity log file. Sun WebServer generates log files using one of three log file formats configurable by the administrator.

httpd.servlet.logs	Logs errors generated by servlets. Log files will be named *prefix.sequence*, where *sequence* is a cycling number.
httpd.site.conf	The sites instance configuration file contains directives that define the site's runtime behavior, and defines the identity and server resources used by a web site hosted by the server instance. This file defines properties such as the web site's canonical host name and aliases, the location of the configuration files, and the network connections available to the web site, and defines the servlet engine settings for the web site.
map.conf	A map directive allows you to redirect requests for a URL on a host to any other URL or to a different directory.
realms.conf	Defines realms of user and group information used by access control lists on a Sun WebServer web site. A realm defines a protection space, a domain of users and groups and their permissions.
servlets.properties	Defines the servlet properties file in a general Java properties file format. The file contains the name of each servlet and the initialization parameters of the servlet.

NAME

access.conf – Defines access control lists that regulate access to web site resources on a Sun ™ WebServer™ web site.

SYNOPSIS

site_path`/conf/access.conf` - Web site access control lists.

`/etc/http/access.conf` - Global access control lists for administration

DESCRIPTION

The `access.conf` file defines Access Control Lists (ACLs) for the content (represented as URI) hosted by a web site. ACLs regulate access to resources on the site by defining which users, groups, and/or hosts have permission to make HTTP GET, PUT, POST, and DELETE requests. ACLs also determine which users, groups, and/or hosts have permission to modify the ACL itself.

ACL definitions may be created by **`htaccess`**(1m), by the Sun WebServer Administration Console, or by Apache Emulator for FrontPage publishing tools.

Note - If FrontPage publishing is active for a site, do not edit the `access.conf` file. FrontPage must be able to read and write the ACL information in a form that it can use, and manual edits may interfere with FrontPage's ability to manage ACL information.

Each ACL definition in the file consists of the following:

- Realm used to store information about valid users.
- Authentication scheme used to ask clients for password information.
- List of users and/or groups in the realm who have permission to change the ACL definition.
- HTTP method block or blocks defining the access permissions on the URI for the given methods.
- Optionally, URI of the resource using the ACL. If the URI is not specified, it applies to all the URIs under this site.
- Optionally, ACL definition blocks for URIs beneath the current one in the document root. These definitions assume the definition of the parent block; explicit directives change the definition in the child block and its children.

Syntax

The following syntax rules apply to the `access.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored.
- White space is ignored in directive definitions.

- A string with a space must be inside double quotes.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last new line with a backslash (\) at the end of the line.
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, new lines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.
- ACL definitions (`url { }` blocks) may be nested within other definitions, as long as the nesting matches the actual URI hierarchy. You cannot nest `url /parent/subdir { ... url /parent { ... } }`. The nested URL is always treated as a path relative to the parent URL.
- The ACL definition on a given URI inherits directives from ACLs on parent URIs, even if the blocks are not nested.

Each ACL definition is in the following form:

```
url <URI> {
  [ realm <realm_name> ]
  [ authentication_scheme <auth_scheme> ]
  [ administrator {
     user <realm_user>[ <realm_user>]...
     [ group <realm_group>[ <realm_group>]... ]
  } ]
  [ method <method_list> {
     [ + | - group * | <realm_group>[ <realm_group>]... ]...
     [ + | - host * | <host>[ <host>]... ]...
     [ + | - user * | <realm_user>[ <realm_user>]... ]...
  } ]...
  [ url <URI> { <ACL> } ]...
}
```

The syntax and definition of each directive and block is explained in the following *Directives* section. Note that all directives are optional. Unless a directive is explicitly defined, its value is inherited from the ACL on the parent URI. If there is no value defined in the parent URI ACL (or any of its parents), the following defaults apply:

`realm`	There is no default. If no `realm` can be defined, then all `user` and `group` directives are ignored.
`authentication_scheme`	If you do not change the default of no `authentication_scheme`, then all `user` and `group` directives are ignored.
`method`	Defaults to `ALL`.

`group`	Defaults to `+ group *`, meaning any authenticated member of any group in the realm is allowed access unless another directive explicitly denies access.
`host`	Defaults to `+ host *`, meaning connections are allowed from any host name.
`user`	Defaults to `+ user *`, meaning any authenticated used defined in the realm is allowed access unless another directive explicitly denies access.
`administrator`	Defaults to either the site or the server administrator. A site administrator is the principal with access to the pseudo-URI "`/sws_administration`" in the site-level `access.conf`. Similarly, a server administrator is a principal with access to the pseudo-URI "`/sws_administration`" in the global `access.conf`.

Directives

The following keyword directives are valid in the `access.conf` file:

`administrator { `*admins*` }` Names users and groups in the specified realm with permissions to change or delete the ACL through the Sun WebServer Administration Console or the **`htaccess`**(1m) utility. The directive *admins* may contain a user directive or optionally a group directive:

`user` *realm_user*`[ `*realm_user*`]` Defines ACL administrators. User names must be valid in the ACL's `realm`.

`group` *realm_group*`[ `*realm_group*`]` Defines groups whose members have permission to change or delete the ACL. Group names must be valid in the ACL's `realm`.

`authentication_scheme basic | md5 | none` Defines the encoding of authentication information for the ACL where `basic` means

user name and password information should be sent in BASE64 encoding over HTTP; `md5` means the server exchanges a message digest of certain header fields and the password is never sent over the wire, and `none` means that no authentication scheme will be used.

`method` *method*`[` *method*`]...` `{` *permissions* `}` Defines blocks of *permissions* that regulate access to the named resource for the HTTP methods listed. The *method* can be `ALL` meaning any method, or it can be a list of one or more of: `DELETE`, `GET`, `POST`, or `PUT`. The *permissions* regulate access to the resource by realm user name, realm group name, or host name:

`+|- host * |` *pattern*	Allows (`+`) or denies (`-`) access to hosts matching the *pattern*. If *host* is an asterisk (`*`), the access permission serves as the default for all hosts. If *pattern* consists only of numbers, it is considered an IP address pattern; otherwise it is considered a host or domain name pattern. For IP addresses, the permission will be applied to any host whose IP address begins with *pattern*. For host or domain names, the permission will be applied to any host whose fully qualified domain name (FQDN) ends with *pattern*.

`+|- group * | `*group*`[ `*group*`]...`	Allows (+) or denies (-) access to authenticated realm users who are members of one of the listed groups. If *group* is an asterisk (`*`), the access permission serves as the default for all groups.
`+|- user * | `*user*`[ `*user*`]..`	Allows (+) or denies (-) access to authenticated realm users. If *user* is an asterisk (`*`), the access permission serves as the default for all users.
`realm` *realm_name*	Defines the realm to use as the source for user name, password, and group information for this ACL. The *realm_name* must exist in the web site's ***realms.conf***(4) file, or all user and group permissions will be ignored.

EXAMPLES

EXAMPLE 1

Access control for the Sun WebServer Administration Console server administration is defined in the global `/etc/http/access.conf` file. The special URI `/sws-administration` defines ACLs for administration:

```
  url "/sws-administration" {
    realm               serverAdmin
    authentication_type basic

    + user  *
  }
```

CODE EXAMPLE 1

A web site ACL to restrict HTTP publishing to valid users:

```
 url "/" {
    realm               publishing
    authentication_type basic

    method PUT DELETE {
```

```
        + user *
    }

}
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htaccess(1m), **htrealm**(1m), **realms.conf**(4)

NOTES

The global `/etc/http/access.conf` is used by the Sun WebServer Administration Server. The command-line utilities protect server-wide administrative access using a pseudo-URI ("`/sws_administration`"). Similarly, the site-wide administrative access is protected by the server pseudo-URI at site specific `access.conf`.

NAME

content.conf – Represents meta-data for directories, site preferences for languages, media and encoding types, and specifying variants for content.

SYNOPSIS

site_path/conf/content.conf

DESCRIPTION

The content definitions may be created by **htcontent**(1m).

Syntax

The following syntax rules apply to the content.conf file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored.
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- Directives of the type name = value require whitespace around the = symbol.
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks. Directive blocks also use matched curly braces to contain a definition. The url blocks can be embedded inside other url blocks. Then the embedded url block inherits information from its parent block unless it explicitly redefines a directive, in which case the directives in the child block will override the directives in the parent.

Content settings take the following form:

```
url <URI> {
variant <variant_info>
 preferences {
  directory_listing {fancy | simple | off}
  default_files
  media_type
  language
  encoding
  charset
 }
 [url <file_uri> {
  filename <variant_info>
  preferences {
    media_type
    language
    encoding
    charset
  }
```

(continued)

(Continuation)

```
 }]...
}
```

The syntax and definition of each directive is explained in the following *Directives* section.

Directives

The following keyword directives are valid in the `content.conf` file:

`preferences` *preferences* Sets the server's content negotiation preferences for the specified URI. Listed in order of preference. Separate multiple preferences with a space. Valid preferences are:

`charset` *charset* [*charset*`...`] Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters. The default `charset` for variants is ISO-8859–1.

`default_file` *filename* [*filename*`...`] Specifies in order of preference the default files Sun WebServer looks for in a directory. If no match is found in this list of file names, the directory contents are displayed. Multiple files must be separated by whitespace. A

different set of default files can be named in any directory.

Use the null string, `""`, as the file name to indicate that there is no default file for the directory. If no file is named, the directory contents will always be listed.

`directory_listing` *method* Sets the method for displaying the contents of a directory when there is no file matching one of the default file names. Listing type can be "`fancy`" to display each directory entry as a hyperlink with the file size, the last modified time, and an icon next to each entry to indicate the file type; "`simple`" to display directory entries as plain text hyperlinks; or "`off`" to display no directory contents.

`encoding` *encoding* [*encoding*...] Specifies the preferred encodings or the encoding type. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals

	which methods have been used to encode the file.
`language` *language* [*language*...]	Specifies in order of preference the preferred languages. Languages are specified in the standard two-letter format.
`media` *media_type* [*media_type*...]	Specifies the preferred types of media. Media type is in standard MIME type format.
`variant_info` *variants*	Sets the server's variants for resources. Separate multiple variants with a space. Valid variants are:
`charset` *charset* [*charset*...]	Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters.
`encoding` *encoding* [*encoding*...]	Specifies the preferred encodings or the encoding type. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals which methods have been used to encode the file.

`language` *language* [*language* ...] Specifies the preferred languages. Languages are specified in the standard two—letter format.

`media_type` *media_type* [*media_type* ...] Specifies the preferred types of media. Media type is in standard MIME type format.

EXAMPLES

EXAMPLE 1

To add an encoded French variant and a German variant for the URI `index.html`:

```
url /index.html {
    index.fr.html lang = fr enc = gzip char = iso-8809-1 type = text/html
    index.de.html lang = de type = text/html; level=3.0
}
```

Note that the `level=3.0` is part of the type definition; it indicates the HTML version level of the resource. If a browser uses the level in a request for an HTML document, and all other things are equal, the server will send the matching resource with the greatest HTML level acceptable to the client.

CODE EXAMPLE 1

To set the default files in a directory to `index.html` or `index.shtml`:

```
url /pages/ {

   preferences {
      directory_listing off
      default_files index.html index.shtml
   }

   url index.html {
      index.ja.html lang = ja char = EUC_JP
      index.en.html lang = en char = iso-8809-1
      index.fr.html lang = fr char = iso-8809-1
   }

}
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO **`htcontent`**(1m)

NAME

httpd-instances.conf – List of all Sun™ WebServer™ server instances.

SYNOPSIS

`/etc/http/httpd-instances.conf`

DESCRIPTION

The `httpd-instances.conf` file associates each unique `httpd` process (or server instance) with its configuration file, and defines whether or not a server instance is enabled. A server instance is a named `httpd` process which services one or more web sites. At installation, there are two default instances: `sws_server` and `admin`, the server process which handles the Sun WebServer Administration Console. The `admin` instance is not created unless `SUNWhtadm` has been installed.

Entries in `httpd-instances.conf` may be modified by using the `htserver` utility, through the Sun WebServer Administration Console, or editing the file directly. It is recommended that you use `htserver` or the Administration Console.

A new server instance is typically needed only when it is impractical to add web sites to an existing server, either because all existing instances already have large numbers of sites or because a new web site requires a configuration that no existing instance can support. Instances cannot share port numbers.

Syntax

The following syntax rules apply to the `httpd-instances.conf` file:

- The exclamation mark '!' is a comment character. All characters from a ! to the end of a line are ignored
- White space is ignored in directive definitions.
- Separate multiple values by white space or tab character.
- Each entry is written on one line with three fields. Additional fields on the same line are ignored.
- There is no syntax error checking for first two fields (*instance-ID* and *configuration_file*). The third field can have any of the following values to indicate that the site is enabled: "enabled", "true", "yes", "1". Any other value in this field would disable the site.

Each entry in `httpd-instances.conf` has the following format

<instance-ID> <configuration_file> <state>

The syntax and definition of each directive are explained in the following *Directives* section.

Directives

The following keyword directives are valid in the `httpd-instances.conf` file:

instance-ID

Shows the unique string used to identify the server instance. The *instance-ID* is used in the Sun WebServer Administration Console, command line utilities, and (by default) to name the configuration file. The string may contain any alphanumeric characters, but may not contain spaces.

The *instance-ID* `'admin'` is reserved to name the Sun WebServer Administration server. You should not delete or rename this server instance unless you want to disable web-based administration.

configuration_file

Shows the full path location of the configuration file for this server instance. By default, the configuration file will be `/etc/http/`*instance-ID*`.httpd.conf`, but any valid configuration file may be used.

state

Shows the state of the server for the purposes of starting all instances (such as when the machine boots). Valid values in this column are:

`enable` Means the server will be started by default when the machine boots or when `htserver start` is run with no explicit instance name.

`disable` Means the server will only be started if it is explicitly named. You must use `htserver start` *instance-ID* to start the instance.

EXAMPLES

EXAMPLE 1

Do not edit the `/etc/http/httpd-instances.conf` file. This example is shown for reference only:

```
Secure_Sites    /etc/http/secure.httpd.conf     disable
Small_Sites     /etc/http/small.httpd.conf      disable
Large_Sites     /etc/http/sws_server.httpd.conf enable
admin   /usr/http/admin_server/conf/admin.httpd.conf   disable
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

htserver(1m)

NOTES

Do not edit `httpd-instances.conf` manually. Use **htserver**(1m) to add, delete, or modify instances.

NAME

httpd.conf – Server instance configuration file.

SYNOPSIS

`/etc/http/`*instance-ID*`.httpd.conf`

DESCRIPTION

The server instance configuration file contains directives that define the server's runtime behavior, the web sites it hosts, and the network connection end points it uses.

The directives are grouped in three major divisions. The valid directives for each division are explained in separate sections below. The major divisions are:

`server { }`	Defines the behavior of the server process. Includes cache settings, the userid of the server process, servlet engine settings, and server-wide defaults for the web sites hosted by the instance.
`url` *hostname* `{ }`	Defines the identity and server resources used by a web site hosted by the server instance. There is one `url {}` block for each web site (or virtual host) handled by the instance, and it defines properties such as the web site's canonical host name and aliases, the location of the configuration file, and the network connections available to the web site.
`port` *number* `{ }`	Defines the settings for a network connection or connections. The server process will attempt to bind to each port on each IP address defined in its `port {}` blocks. If a port is in use, the server will log an error. The server will exit if it cannot bind to at least one port. Port directives include the IP addresses on which to use the port and the timeout period.

Syntax

The following syntax rules apply to the `httpd.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- Any value may optionally be enclosed in double quotes (").

- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.

The overall format of the file is:

```
server {
   <server directives>
}
url <hostname> {
   site_path <absolute_pathname>
   site_conf <filename>
   <web site directives>
}
[ url <hostname> { <web site directives> } ]...
port <number> {
   <port directives>
}
[ port <number> { <port directives> } ]...
```

There must be only one `server {}` block. There must be at least one `url {}` block, and each block–except for the default site—must have a unique *hostname*, a `site_path` directory that exists, and a `site_conf` file that exists in the `site_path`. There must be at least one `port {}` block, and duplicate port numbers are allowed as long as the IP address definitions are different.

See the *Extended Description* for the syntax of all valid directives in each division.

EXTENDED DESCRIPTION

`server {}` Block Directives

`access_enable yes | no` Enables or disables the use of access control lists (ACLs) to control access to resources on the server. The default is `yes`. If ACLs are enabled, the server will look for ACLs in the file *site_path*`/conf/access.conf` for each web site hosted by the server instance.

`cache_enable yes | no` Enables or disables server-side caching of documents. The default is `yes`. Caching can improve the performance by avoiding file system accesses for frequently requested static documents. Dynamic content, such as CGI output, is not cached.

`cache_large_file_cache_size` *MB*
: Sets the size in MB of the server-side cache for large files. The large file cache caches files larger than 16 KB. The largest file size cached can be set with `cache_max_file_size`. The default value is 256 MB, and the maximum is only limited by virtual address space of the Sun™ WebServer™ process. You should rarely need to adjust this parameter.

`cache_max_file_size` *MB*
: Sets the size of the largest file that will be cached in the large file cache. Documents that are larger than `cache_max_file_size` will never be cached. Caching works best by serving a large number of frequently requested documents. If the cache is filled with only a few very large documents, caching performance will be poor. The default `cache_max_file_size` is 1 (MB).

`cache_small_file_cache_size` *MB*
: Sets the size of the smallest file that will be cached in the small file cache. Documents smaller than `cache_small_file_cache_size` will never be cached. The small file cache contains only documents smaller than 16 KB. Larger documents are stored in the large file cache or not cached at all. The default `cache_small_file_cache_size` is 8 (MB).

`cache_verification_time` *seconds*
: Sets the number of seconds the server will wait before verifying the validity of a cached file. When Sun WebServer retrieves a file from the cache, if it has not been verified in `cache_verification_time` seconds, it is re-verified. If the dates are different, the file is removed from the cache and the actual file is retrieved. Cached files may become out of date if the actual file has changed. Cached files are verified by checking the date stamp of the cached file against the date stamp of the actual file. If your files only rarely change, you may want a higher value for `cache_verification_time`. The default is 10 (seconds).

`cgi_error_log_cycle_time` *minutes*
: Sets the maximum age, in minutes, for CGI error log files for the current host. If a CGI error log is older than the number of

`cgi_error_log_cycle_time` minutes, then a new request log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is 1440 minutes (1 day).

`cgi_error_log_enable yes|no` Enables or disables CGI script error logging. The default is `no`.

`cgi_error_log_max_files` *integer* Sets the maximum number of CGI error log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `cgi_error_log_max_files`, it is reset to 1 and the first CGI error log file is overwritten. This prevents the number of log files from growing without limit. Set `cgi_error_log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`cgi_error_log_max_file_size` *bytes* Sets the maximum file size, in bytes, for CGI error log files for the current host. If a CGI error log exceeds `cgi_error_log_max_file_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`cgi_error_log_prefix` *prefix* Sets the directory and log file name prefix for cgi script error logs for a given host. The string must have an absolute path name followed by the prefix that will be used for this host's log files. Separate virtual hosts must have different prefix names so that there is no conflict in writing to the logs. Each of a hosts log files will have a name in the form:

`<absolute pathname>/prefix.<sequence>`

For example, `/var/http/logs/http.2`. Sequence is incremented and a new file created whenever `cgi_error_log_cycle_time` or `egi_error_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is "`var/http/logs/http`".

`comment "`*description*`"`

Simply a comment string to describe the current configuration file.

`default_file` *file* [*file*...]

Lists in order of preference the name of the file Sun WebServer will look for in a directory when a URL request does not name a specific file. For example, if the URL request is for `http://hostname/`, Sun WebServer will look at the top directory of host name's doc_root for a file named *default_file*.

If `default_file` does not appear in the configuration file, then "index.html" is used. If `default_file` is " ", then no default file is used. If multiple files are specified, then the files are used in the specified order.

If no file matching the values for `default_file` is found, the directory contents will be listed, subject to the value of `directory_listing`.

The built in server-wide default is "`index.html`".

`directory_listing fancy | simple | off`

Specifies how the contents of directories will be listed if no file matching default_file is found in the directory. `directory_listing` is valid in server {} blocks as server-wide defaults or in url {} blocks for per-host settings.

`fancy`	Displays directory contents with each name as a hyperlink to the file, icons matching each file's type, and file size and date information. The icons used and the association of icons to file suffixes is configurable.
`simple`	Displays only each file name as a

		hyperlink to the file itself.
	`off`	Disables displaying directory contents; an HTTP "404 Not Found" error is returned to the client instead.

The built in server-wide default is “`fancy`”.

`error_document` *http_error_code url*

Allows customized error messages to be returned to the client. Any valid URL, including CGI scripts, may be returned, so you have flexibility in what information you want to give clients when an error occurs. When an HTTP error code is returned, Sun WebServer will return an HTTP "302: Document Moved" header with a Location: header indicating the file to which the error has been remapped. Most clients will automatically fetch the URL named by the Location: header. You can redirect the following HTTP error codes:

- 400 - Bad Request (Remapped by default host only)
- 403 - Forbidden
- 404 - Not Found
- 412 - Precondition Failed
- 500 - Server Error
- 501 - Not Implemented
- 503 - Service Unavailable

The destination URL can be relative to the current or default host if it begins with a "/". Otherwise, an absolute URL must be specified. The following examples show a relative and absolute URL, respectively:

- error_document "503"
- "/cgi-bin/error.pl?503"
- error_document "500"
- "http://www2.A.com/mirror/"

Note - The path to which you remap 404 errors must be available in the document root. You can not redirect 404 errors to aliased directories. `error_document` is valid in server {} blocks as server-wide defaults or in url {} blocks for per-host settings.

`icon_add` *alt_text bitmap_URI file_type*`[`*file_type*`]` Allows associating a file extension with an icon file. The icon will be used to represent all files with the extension in `fancy` directory listings. `icon_add` can also change a default association or assign icons to file types not covered in the default set; for example, you may want to use your own icons to represent basic types. In addition to file suffixes, you can customize the icon used for "parent directory" (`../`) and "subdirectory" by specifying "UP" or "DIR" as the *alt_text*.

alt_text	Specifies a string that will be used instead of an icon in text-only browsers. For example, "`GIF`".
bitmap_URI	Specifies a URI path relative to the default host (in the server {} block) or the host named by the current `url {}` block. For example, "`/icons/binary.xbm`".
file_suffix(es) **|** ***content_type(s)***	Specifies a string of one or more file extensions or content type definitions that will use the icon in "`fancy`" directory listings. For example, "jpg jpeg JPG image/jpeg"

assigns the icon to all files ending in .jpg, .jpeg, and .JPG or files with a MIME type of image/jpeg.

In addition to file suffixes, you can customize the icon used for "parent directory" (`../`) and "subdirectory" by specifying "UP" or "DIR" as the *alt text*. `icon_add` is valid in server {} blocks as server-wide defaults or in url {} blocks for per-host settings.

`icon_default` *bitmap_URI* Sets the icon used for files with extensions that do not have a defined icon type. The path to the bitmap file must begin with a "`/`", and it is relative to the default host or the host defined by the current url {} block. `icon_default` is valid in server {} blocks as server-wide defaults or in url {} blocks for per-host settings.

`lwp_threads_count` *integer* Sets the total number of lightweight process (LWPs) threads in the kernel that Sun WebServer will attempt to use to map to user threads in its thread pool. By default, Sun WebServer uses 1 LWP per user thread (lwp_threads_count will equal threads_n_active).

Increasing this number may increase the actual thread concurrency of Sun WebServer, and it will also increase the share of system resources used by the httpd process.

`mime_file` *relative_path* Contains the default MIME types definitions used for the server host. Default is `/etc/http/mime.types`. The order in which Sun WebServer will attempt to match a file with a MIME type is as follows:

1. Check any host-specific `mime_add` definitions.
2. Check any `mime_add` and `mime_file` definitions in the `server {}` block.

3. If none of the above produces a MIME type for the file, the `mime_default_type` is used. Entries in the mime_file have the form:

   `<type>/<subtype><file suffix(es)>`

   For example: text/html html htm. The default server-wide `mime_file` is `/etc/http/mime.types`.

Note - If `mime_file` and `mime_add` directives appear in the server block, later definitions will replace earlier definitions. For example, if a `mime_add` definition appears after the `mime_file` directive, it will replace a matching definition in the MIME types file. `mime_file` is valid in server{} blocks as server-wide defaults or in url {} blocks for per-host settings.

`mime_default_type` *type/subtype* Sets the MIME type that will be used for files whose extension does not match any other MIME type. The default is "text/html". `mime_default_type` is valid in server{} blocks as server-wide defaults or in url {} blocks for per-host settings.

`server_admin` *email_address* Specifies the email address of the Sun WebServer administrator.

`server_classpath` *path*`[:`*path*`]...` Specifies the Java™ classpath where the JDK, JSDK, and classes for the servlet engine are located.

Note - This is not the path for servlets to be loaded.

`server_root` *absolute_path* Specifies the path in which the server-wide scripts, icons, and base for sites hosted by the instance are stored.

`server_user` *username* Sets the user name Sun WebServer will use after start up. The string must be a valid user name on the system. After httpd is started by root, it will change to the server_user user name. The default server_user is root. The server_user must be able

to read and write the configuration file, any ACL files, read files in the doc_root directories, and be able to read and write files in any log file directories. You can set the server_user so the server does not run with root permissions. By running as a different user, there are fewer security risks because the server will be unable to change or serve to clients sensitive files owned by root. You may want to create a user only to run Sun WebServer, and make sure that this user owns or has access to all of the necessary files and directories.

`servlet_engine { `*parameters*` }` Specifies parameters of the servlet engine:

`chaining_enable yes | no` Enables or disables servlet chaining. This enables the servlet engine to run a sequence of servlets in a specified order to fulfill one single servlet request. Host administrators can specify a chain of servlets to be executed sequentially. Default is "`no`".

`dynamic_linking_enable all | local | remote | none` Allows the specified servlets access to dynamic libraries. Default is "none".

`file_access_enable all | local | remote | none` *relative_path* Allows specified servlets to have access to file resources, for example, read / write a file on local disk. Default is "`local`".

`network_access_enable all | local | remote | none` Allows specified servlets to have access to network

resources, for example, open a socket. Default is "`local`".

`properties_file` *relative_path* Path to "`servlets.properties`". All loadable servlets are specified in this file. Each servlet engine instance can have a list of preloaded servlets that it wishes to load and initialize as soon as the server starts. This list is specified in a servlet properties file. Path can be either absolute or relative to `site_path`.

`reload_enable yes | no` Allows or disallows the servlets to reload. If the servlet classfile changes, a servlet instance (reflecting the changes) can be reloaded. Reloading can be performed by either the server or the host administrator, depending on whether the hosts are sharing a servlet engine instance. Servlets do not reload automatically and will reload only when explicitly requested. The default is "`no`".

`remote_enable yes | no` Enables or disables remote servlets. Enabling remote servlets allows the servlets from remote sites to be loaded by the server. The default is “`yes`”.

`se_log_enable yes | no` Enables or disables the servlet error logging. The default is “`no`”.

`se_log_cycle_time` *minutes* Sets the maximum age, in minutes, for servlet error log files for the current host. If a servlet error log is older than the number of `se_log_cycle_time` minutes, then a new servlet error log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is 1440 minutes (1 day).

`se_log_max_files` *integer* Sets the maximum number of servlet error log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a

new log file is created. If the sequence goes beyond `se_log_max_files`, it is reset to 1 and the first servlet error log file is overwritten. This prevents the number of log files from growing without limit. Set `se_log_max_files` to “-1” to have no limit on the number of log files. Default is 7 files.

`se_log_max_file_size` *bytes* Sets the maximum file size, in bytes, for servlet error log files for the current host. If a servlet error log exceeds `se_log_max_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`se_log_prefix` *absolute_path/prefix* Specifies the directory and log file name prefix for servlet error logs for a given host. The string must have an absolute path name followed by the prefix that will be used for this host’s servlet log files. Separate virtual hosts must have

different prefix names so that there is no conflict in writing to the logs. Each host's servlet log files will have a name in the form:

`<absolute pathname>/prefix.<sequence>`

For example, `site_path/logs/se_logs.2`. Sequence is incremented and a new file created whenever `se_log_cycle_time` or `se_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is"`site_path/logs/se_logs`".

`security_access_enable all | local | remote | none` *absolute_pati*

Allows the specified servlets to have access to security resources, for example, classLoaders. Default is "`local`".

`system_access_enable all | local | remote | none`

Allows or disallows the specified servlets to have access to system resources, for example, call `System.Exec()`. Default is "`local`".

`servlets_path` *path* — Specifies the absolute path to local directories for jar files for all local servlets. This is a colon-separated list.

`server_servlets_enable` `yes | no` — Allows or disallows servlets to run in this server process. This option controls whether to start the JVM for this process. The default is "`no`".

`site_restrictions { directives }` — Enables access to the following:

`cgi_superuser yes | no` — Allows or disallows cgi_user setting of any web site to be root. Default is "`no`".

`cgi_user_unique yes | no` — Determines whether the cgi_user setting of any web site must be unique throughout the server. Default is "`yes`".

`cgi_dns_enable yes | no` — Enables or disables the `REMOTE_HOST` `CGI` environment variable to be set and to be available to CGI scripts. `REMOTE_HOST` requires a DNS lookup of the IP address (`REMOTE_ADDR`) of the resource making the CGI request. Since DNS lookups can be resource consuming, allowing such DNS lookups can slow

		performance, especially on a server that uses extensive CGI. The default is "no".
	se_share yes | no	Enables all web sites to share the servlet engine defined in the server block if set to "yes". If set to "no", there is no sharing and each web site can have its own servlet engine. This is a server-wide setting. The default is "no".
symlink_follow yes | no		Follows or ignores symbolic links in the file system. Ignoring symbolic links may cause a performance loss as the file name and each directory in the path of a requested resource must be checked to make sure there are no symbolic links. Following symbolic links may be a security risk because a symbolic link can potentially point to a file that is outside of the doc_root. A symbolic link to a sensitive file (such as /etc/passwd) can only be made by someone with write access to the file, so the security risk is often small and easily managed by controlling who has access to the document root. symlink_follow is valid in server {} blocks as server-wide defaults or in url {} blocks for per-host settings. Default is "yes".
threads_n_active *integer*		Specifies the maximum number of user threads Sun WebServer will have available in its thread pool. The number of threads will not grow beyond this number. Sun WebServer uses one thread per connection, releasing the thread to the thread pool when the request has completed. For keepalive connections, the thread is released to the thread pool after a request has completed, and a new thread is used if there is a new request on the connection. The number of threads sets an

upper limit on the number of simultaneous connections Sun WebServer can handle. The default value is 128 threads.

`user_doc_source` *source* — Specifies the source of user information for user document directories if `user_doc_enable` is "yes." In most cases, the only valid value is UNIXSYS, and users are defined through the operating system (for example, in `/etc/passwd` or NIS).

In Solaris for ISPs, if virtual FTP servers have been defined in Sun Directory Services, you may set this to `ISP`. User information will be taken from the directory server, and the value of a user's `ispContentDirectory` will be used.

`version` *version_string* — Describes the current version of Sun WebServer.

`url {}` Block Directives

`alias` *hostname* — Defines other names for the specified virtual host.

`conn_end_points [`*IP_address*`|*`*`]:`*`port`* — Determines the IP addresses and port numbers on which requests to this web site are accepted. The syntax of this directive is

```
[<ip_address>]:port number
```

If the ip_address refers to the set of all IP addresses for this web site, then for HTTP 1.1 virtual hosts, leaving the ip_address field blank means that it is available on all IP addresses.

This directive may not appear in the url {} block for the default virtual host. In consequence, the default host will receive all requests to unknown hosts and all HTTP 1.0 requests not addressed to any host that arrive on any port. The server rejects with error 400 HTTP 1.1 requests not addressed to any host.

`owner_group` *groupname* — Defines the UNIX group who owns the web site's content files.

`owner_user` *username* — Defines the UNIX user and group who own the web site's content files.

	`site_config` *path*	Specifies the relative path configuration file for the web site.
	`site_enable yes | no`	Determines whether the site is currently enabled and accepting requests. `site_enable` is not an initialized setting. A site is enabled once it has been configured successfully. After successful configuration, use `hthost` to enable or disable the site.
	`site_path` *directory_path*	Specifies the absolute path location of the web site.
`port {}` **Block Directives**		
	`ip_address` *nnn.nnn.nnn.nnn*	Indicates the IP address on the server that can receive requests on the current port. Use this if you do not wish to support all IP addresses on a port. You need to create a separate port {} block with the same port number for each specific ip_address you want to support. Use separate port {} blocks with unique ip_address settings to support IP-based virtual hosting. If this parameter is omitted, all IP addresses on the server will be supported on the port. The default is all IP addresses.
	`keepalive_enable yes | no`	Allows or disables HTTP 1.0 keepalive connections on the current port. HTTP 1.1 connections always use keepalive, but HTTP 1.0 browsers can only establish a keepalive connection with Sun WebServer by sending a Connection: keepalive HTTP header (if keepalive_enable is set). Keepalive may improve performance since the connection is not destroyed and reestablished for each HTTP request. The default is "yes".
	`request_timeout` *seconds*	Sets the maximum time, in seconds, that Sun WebServer will wait to fill an individual client request on the current port. The default is 180 seconds (three minutes).
	`ssl_client_cert_required yes | no`	Determines whether the server will demand a certificate signed by a CA known to the server when a client connects to the SSL port. The default is "no".

`ssl_enabled yes | no` Enables or disables the Secure Sockets Layer (SSL). SSL encrypts and authenticates messages sent between a browser and Sun WebServer. Encryption using public key cryptography ensures the privacy of the messages sent between the client and Sun WebServer. Port 443 is the default SSL port and is recommended for easiest use by clients (no port will need to be specified in the https URL). The default is "no".

Note - To run SSL, you will need to set up a local root Certificate Authority (CA) with a Distinguished Name record, and generate a private/public key pair for the local root CA. The local root CA will be able to generate credentials and key pairs for every SSL-enabled host within your organization, whether it is a single host or a machine running Sun WebServer with hundreds of virtual hosts. The security tools use the Federated Naming System (FNS) to manage the naming context for users and hosts with certificates.

`ssl_ciphers` *cipher_string(s)* Sets the cipher parameters used for SSL encryption. It may be one of the following:

SSL_RSA_EXPORT_WITH_RC4_40_MD5 RC4 40–bit exportable ciphers. This is the default setting.

SSL_RSA_WITH_RC4_128_MD5 RC4 128–bit, North America only cipher (this requires separate SSL packages not available in the downloadable version of Sun WebServer).

EXAMPLES

EXAMPLE 1

A sample configuration file:

```
# Automatically generated.
#
#       SWS HTTP server configuration file
#
# file version number
version "SWS2.0"

server {
        comment                 "Sample Configuration"
        server_root             "/var/http/sample/"
        server_user             "root"
        cache_enable            "yes"
        cache_large_file_cache_size 256
        cache_max_file_size     1
        cache_small_file_cache_size 8
        cache_verification_time  10
        cgi_error_log_cycle_time 1440
        cgi_error_log_enable    "no"
        cgi_error_log_max_files  7
        cgi_error_log_max_file_size 1048576
        cgi_error_log_prefix    "error_log"
        lwp_threads_count       1
        threads_n_active        128
        access_enable           "yes"
        symlink_follow          "yes"
        user_doc_source         "ISP"
        directory_listing       "fancy"
        default_file            index.html
        mime_default_type       "text/html"
        mime_file               "/etc/http/mime.types"
        icon_default            "/sws-icons/unknown.xbm"

        site_restrictions {
                se_share                "yes"
                cgi_superuser           "no"
                cgi_user_unique         "no"
                cgi_dns_enable          "no"
        }
        server_classpath        "/usr/lib/http/classes.zip:/usr/jsdk/lib/classes.zip:/usr/java/li
        server_servlets_enable  "yes"
}

url {
        site_enable             "yes"
        site_path               "/var/http/sample/websites/default_sites"
        site_config             "conf/default_sites.site.conf"
        owner_user              "admin3"
}

port 80 {
        ip_address          129.146.146.146
        keepalive_enable        "yes"
        request_timeout         180
}
```

ATTRIBUTES See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO **htserver**(1m), **hthost**(1m), **httpd.site.conf**(4)

NAME

httpd.event.logs – Description of the Events Log files.

SYNOPSIS

`/var/adm/messages`

DESCRIPTION

The httpd server puts out error messages and warnings via **`syslogd`**(1m) to `/var/adm/messages` by default. Use `/etc/syslog.conf` to change the default location.

Syntax

The definitions in the `httpd.event.logs` file have the following format:

<time> <host> sws.<instance>[pid]: [<version> <message_id> <seq_no> (<source>) <severity>]: <message>

Errors sent to screen before the daemon has been created or started are sent in console format. The format of the messages written to the console will be:

<message_id> <time> <severity>: <message>

The following keyword directives are valid in the `httpd.event.logs` file:

time The date and time (in the format: MMM DD hh:mm:ss) that the error occurred.

host The node name of the host (`uname ---n`).

tag [pid] A tag and a pid form the framework for error messages from user space. The tag reveals which Sun™ WebServer™ daemon logged the message. The instance name spans multiple starts and stops of a single server. The tag contains the instance name of the server preceeded by “sws”. The pid contains the process id of the process that generated the error message.

version The version of the error message format.

message_id The unique identifier for the error message with the form *modulename.id_number*. The *modulename* refers to a shared object or to a subsystem in the httpd daemon code. The *id_number* mirrors the message catalog number.

seq_no A sequence number determines the exact sequence of error messages in the log. This field is always “0” for Sun WebServer.

source This field contains the source of the error. The source is always “SW” for Sun WebServer.

severity The errors can be one of four severity levels:

CRITICAL	Critical condition such as failure to daemonize.
ERROR	Error conditions such as server being unable to bind to port. Error conditions cause a significant part of the server to fail.
WARNING	Warning conditions such as being unable to write to a log file. Warning conditions do not prevent the server from running.
NOTICE	Normal but significant conditions. May require special handling.

message The actual text of the message.

EXAMPLES

EXAMPLE 1

Sample event log entries:

```
May 17 15:39:10 hostname sws.admin[14454]: [1 httpd.134 0 (SW) NOTICE]: Received a SIGHUP signal;

May 17 15:39:10 hostname sws.admin[14454]: [1 admin.195 0 (SW) NOTICE]: Running with SWS Configura

May 18 15:35:33 hostname sws.Secure_Sites[15257]: [1 net.61 0 (SW) WARNING]: Network interface 129
```

```
May 18 15:35:33 hostname sws.Secure_Sites[15257]: [1 net.183 0 (SW) ERR]: httpd cannot bind to any
May 18 15:36:49 hostname sws.Large_Sites[15077]: [1 servlet.231 0 (SW) WARNING]: Servlets not allow
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

syslogd(1m), **syslog.conf**(4)

NOTES

To ensure that all Sun WebServer messages are logged by **syslogd**(1m), create an entry for `daemon.notice` messages in **syslog.conf**(4).

For example, to log Sun WebServer messages to `/var/adm/messages` create the following entry:

daemon.notice /var/adm/messages

Note - The whitespace between the message type and the file name consists only of tabs.

This entry would cause all messages with a severity of “notice” or greater generated by any daemon process to be logged to `/var/adm/messages`.

NAME

httpd.request.logs – Description of the Sun™ WebServer™ request log files.

SYNOPSIS

site_path/log-prefix

DESCRIPTION

The httpd.request.logs file logs all incoming requests to a server request log file. Sun WebServer generates log files using one of three log file formats configurable by the administrator: the Common Log Format (CLF), the Extended Common Log Format (ECLF) also known as the Combined Log Format, or the Extended Log Format (ELF). The location of a request log file is set by the log-prefix directive in the *site_path/conf/httpd.site.conf* or in the Sun WebServer Administration Console.

The most recent log file can be accessed through the log prefix. If the *prefix* is *site_path/default_log* and the log type is ELF, then apart from having log files with names */site_path/default.log.elf.1*, you will also have a symbolic link called *site_path/default.log.elf*, which points to the latest log file that the server is writing.

Common Log Format (CLF)

The CLF format is used by most HTTP servers and analysis tools.

To get CLF logs, set `log_type` to `''clf''` in the web site configuration file.

The entries in the Common Log File (CLF) logs have the following format:

```
host rfc931 authuser [DD/Mon/YYYY:hh:mm:ss] "request" ddd bbbb
```

The following list explains the log entry fields::

host	The DNS name or the IP number of the remote client.
rfc931	The information returned on this client for this request, otherwise (-).
authuser	The userid sent for authentication, otherwise (-).
DD/Mon/YYYY:hh:mm:ss	The date and time of the request.
request	The first line of the HTTP request as sent by the client.
ddd	The HTTP status code returned by the server, if not available (-).
bbbb	The number of bytes sent, not including the HTTP header, if not available (-).

Extended Common Log Format (ECLF)

An extended common log format file is a variant of the common log format file with two additional fields at the end of the line, the referrer and the user agent fields.

To get ECLF logs, set `log_type` to `''eclf''` in the web site configuration file.

The entries in the ECLF log files have the following format:

<host> rfc931 authuser [DD/Mon/YYYY:hh:mm:ss GMT_offset] "request" ddd bbbb "referrer" "user_agent"

The following list explains the log entry fields:

host	The DNS name or the IP number of the remote client.
rfc931	The information returned on this client by `identd` for this request, otherwise (-).
authuser	The userid sent for authentication, otherwise “-”.
DD/Mon/YYYY:hh:mm:ss	The date and time of the request.
GMT_offset	The difference between the local time and the Greenwich Mean Time (GMT).
request	The first line of the HTTP request as sent by the client.
ddd	The status code returned by the server, if not available (-).
bbbb	The number of bytes sent, not including the HTTP header, if not available (-).
referrer	The url the client was on before requesting your url, if not available, (-).
useragent	The software the client claims to be using, if not available (-).

Extended Log Format (ELF)

The Extended Log File (ELF) format is a flexible format for recording HTTP requests, which is particularly suited for log analysis tools. ELF records more information than the CLF format. It contains a sequence of lines containing ASCII characters delimited by a new line. Lines that start with # are comment directives.

To get ELF logs, set `log_type` to `''elf''` in the web site configuration file.

The entries in the ELF log files have the following format:

date time cs-method cs-uri sc-status time-taken bytes cs-ip cs-host

The following list explains the log entry fields:

date	The date in yyyy-mm-dd format.
time	The time the request was accepted by the server in the 24–hour format in the machine's time zone.
cs-method	The client-to-server HTTP method.
cs-uri	The client-to-server requested URI.
sc-status	The server-to-client HTTP status code in the response.
time-taken	The time taken to process the request.
bytes	The number of bytes sent.
cs-ip	The client-to-server IP address and port of the client.
cs-host	The DNS name of the remote client, if not available "-".

EXAMPLES

EXAMPLE 1

Common log format (clf):

```
129.146.114.74 - - [18/May/1998:17:47:58 -0700] "GET /monhelp/SISP_Banner.html HTTP/1.0" 200 558
129.146.114.74 - - [18/May/1998:17:48:11 -0700] "GET /swshelp/ht_server_cache_settings.html HTTP/1
129.146.114.74 - - [18/May/1998:17:48:12 -0700] "GET /swshelp/ht_server_cache_settings1.html HTTP/
129.146.114.74 - - [18/May/1998:17:48:12 -0700] "GET /swshelp/SWS_Banner.html HTTP/1.0" 200 330
129.146.114.74 - - [18/May/1998:17:48:25 -0700] "GET /swshelp/ht_server_web_sites_list.html HTTP/1
129.146.114.74 - - [18/May/1998:17:49:09 -0700] "GET /mchelp/awsTOC.doc.html HTTP/1.0" 200 5953
129.146.114.74 - - [18/May/1998:17:48:12 -0700] "GET /swshelp/SWSbanner.gif HTTP/1.0" 200 16712
```

CODE EXAMPLE 1

Extended log format (elf):

```
#Version: 1.0
#Software: Sun_WebServer/2.0
#Start-Date: 1998-05-18 17:57:20
```

```
#Fields: date time cs-method cs-uri sc-status time-taken bytes cs-ip cs-host cs-referer cs-agent
1998-05-18 17:57:01 GET / 200 0 848 129.146.114.74:49028 -
1998-05-18 17:57:13 GET /swshelp/ht_helpTOC.doc.html 304 0 142 129.146.114.74:49028 -
1998-05-18 17:57:20 GET /swshelp/ht_server_cache_settings.html 304 0 142 129.146.114.74:49028 -
1998-05-18 17:57:21 GET /swshelp/frame.ht_server_cache_settings1.html 304 1 142 129.146.114.74:4903
1998-05-18 17:57:21 GET /swshelp/SWSbanner.gif 304 0 142 129.146.114.74:49032 -
```

CODE EXAMPLE 2

Extended common log format (eclf):

```
129.146.114.74 - - [18/May/1998:18:02:56 -0700] "GET /swshelp/ht_server_web_sites_list.html HTTP/1.
://isp-doc1/swshelp/ht_helpTOC.doc.html" "Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
129.146.114.74 - - [18/May/1998:18:03:03 -0700] "GET /swshelp/ht_server_iplist2.html HTTP/1.0" 200
-doc1/swshelp/ht_helpTOC.doc.html" "Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
129.146.114.74 - - [18/May/1998:18:03:05 -0700] "GET /swshelp/ht_server_iplist3.html HTTP/1.0" 200
doc1/swshelp/ht_helpTOC.doc.html" "Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
129.146.114.74 - - [18/May/1998:18:03:02 -0700] "GET /swshelp/frame.ht_server_iplist2.html HTTP/1.0
//isp-doc1/swshelp/ht_helpTOC.doc.html" "Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Stable

SEE ALSO

httpd.site.conf(4)

NAME

httpd.servlet.logs & httpd.cgi.logs – The log files for servlet requests and CGI script errors.

DESCRIPTION

The `httpd.servlet.logs` log errors are generated by servlets. You can specify the location of the log files with `htservlet` or through the Sun™ WebServer™ Administration Console. See **`htservlet`**(1m) for details on configuring log settings.

The `httpd.cgi.logs` log errors are generated by cgi scripts. You can specify the location of the log files through the Sun WebServer Administration Console.

Log files will be named *prefix.sequence*, where *sequence* is a cycling number. Each entry is a Common Log File (CLF) format header indicating the request that launched the servlet or CGI, followed by all error messages generated during the request.

The servlet error logs will have the following format:

{client_name [DD/Mon/YYYY:hh:mm:ss] "request" "vhostname" "script_file_name"}
error_messages

where *error_messages* is the error output for the servlet.

The CGI error logs will have the following format:

{client_name [DD/Mon/YYYY:hh:mm:ss] "request" "vhostname" "script_file_name" [block_number]}
error_messages

where *error_messages* is the error output for the CGI script, and *block_number* identifies the block number of the CGI script. The final block from a script also has a keyword “FINAL” at the end of the header. Since Sun WebServer uses constant buffer sizes and many CGI scripts are using the same error log file, it is impossible that error messages from a given script are logged into more than one of the above blocks. If Sun WebServer could log all the error messages from the script into just one block, then it doesn’t print either the block number of the keyword “FINAL”.

The most recent log file can be accessed through the log prefix. If the *prefix* is `/var/http/default_log`, then apart from having log files with names `/var/http/default_log.1`, you will also have a symbolic link called `/var/http/default_log`, which points to the latest log file that the server is writing.

See **`httpd.request.logs`**(4) for a description of the CLF.

EXAMPLES

EXAMPLE 1

Sample CGI error logs:

```
{129.146.115.80 [14/Mar/1998:11:03:23 -0800] "GET /cgi-bin/login HTTP/1.0" www.A.com /tmp/tp/root1/
Invalid number of parameters

{129.146.115.80 [14/Mar/1998:12:03:29 -0800] "GET /cgi-bin/sendfile HTTP/1.0" www.A.com /tmp/tp/roo
Unable to open file:
    /var/http/server/websites/public/profiles/file.1

Output not generated.

{129.146.115.80 [14/Mar/1998:13:03:35 -0800] "GET /cgi-bin/login HTTP/1.0" www.A.com /tmp/tp/root1/
Authentication failed for user "user1"
```

CODE EXAMPLE 1

Sample servlet error logs:

```
{129.146.114.181 [19/May/1998:10:21:21 -0700] "GET /servlets/network HTTP/1.0" www.A.com /servlets/
cannot open url connection: connect

{129.146.114.181 [19/May/1998:10:21:21 -0700] "GET /servlets/network HTTP/1.0" www.A.com /servlets/

java.lang.NullPointerException
 at com.sun.sws.se.ServletEngine.newServlet(Compiled Code)
 at com.sun.sws.se.ServletEngine.loadLocalServlet(Compiled Code)
 at com.sun.sws.se.ServletEngine.loadServlet(Compiled Code)
 at com.sun.sws.se.ServletEngine.loadAndInitServlet(Compiled Code)
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

httpd.conf(4), **httpd.site.conf**(4), **httpd.request.logs**(4)

NAME

httpd.site.conf – Site instance configuration file.

SYNOPSIS

site-path`/conf/`*site_name*`.site.conf`

DESCRIPTION

The site instance configuration file contains directives that define the site's runtime behavior, and defines the identity and server resources used by a web site hosted by the server instance. This file defines properties such as the web site's canonical host name and aliases, the location of the configuration file, and the network connections available to the web site. It also defines the servlet engine settings for the web site. The location of `httpd.site.conf` is determined by the site_config directive in `httpd.conf`.

Syntax

The following syntax rules apply to the `httpd.site.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- Any value may optionally be enclosed in double quotes (").
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.

The overall format of the file is a list of directives.

See the *Extended Description* for the syntax of all valid directives in each division.

EXTENDED DESCRIPTION

Directives

`access_file`	Specifies the file in which the server will look for ACLs for each web site hosted by the server instance. The default is *site_path*`/conf/access.conf`.
`afp_enable yes | no`	Enables Microsoft FrontPage support.
`cache_control public | private | no cache`	Sets the caching policy used by intermediate caching proxies. The "`public`" setting allows the cache to be shared with other proxies. The

	"private" setting allows the files to be cached, but the cache can not be shared with other proxies. The "no_cache" setting does not allow the cache to be shared with other proxies. The default is "public."
cgi_dns_enable yes | no	Enables or disables the REMOTE_HOST CGI environment variable to be set and to be available to CGI scripts. REMOTE_HOST requires a DNS lookup of the IP address (REMOTE_ADDR) of the resource making the CGI request. Since DNS lookups can be resource consuming, allowing such DNS lookups can slow performance, especially on a server that uses extensive CGI. The default is no.
cgi_enable yes | no	Enables or disables the authority to execute CGI scripts for the host defined by the url {} block. If enabled, resources in the CGI directory can be accessed through URLs of the form http://hostname/cgi-bin/ and executed as CGI scripts. If enabled and cgi_suffix_enable is also enabled, then any file with .cgi extension can be executed as a CGI script. Individual sites can override this server setting. The default is no.
cgi_error_log_cycle_time *minutes*	Sets *minutes* as the maximum age, in minutes, for CGI error log files for the current host. If a CGI error log is older than the number of cgi_error_log_cycle_time minutes, then a new request log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is 1440 minutes (1 day).
cgi_error_log_enable yes | no	Enables or disables CGI script error logging. The default is no.
cgi_error_log_max_files *integer*	Sets *integer* as the maximum number of CGI error log files that Sun™ WebServer™ will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond cgi_error_log_max_files, it is reset to 1 and

the first CGI error log file is overwritten. This prevents the number of log files from growing without limit. Set `cgi_error_log_max_files` to “-1” to have no limit on the number of log files. Default is 7 files.

`cgi_error_log_max_file_size` *bytes* Sets the maximum file size, in bytes, for CGI error log files for the current host. If a CGI error log exceeds `cgi_error_log_max_file_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`cgi_error_log_prefix` *prefix* Sets the directory and log file name prefix for CGI script error logs for a given host. The string must have an absolute path name followed by the prefix that will be used for this host’s log files. Separate virtual hosts must have different prefix names so that there is no conflict in writing to the logs. Each host’s log files will have a name in the form:

`<absolute pathname>/prefix.<sequence>`

For example, `/var/http/logs/http.2`. Sequence is incremented and a new file created whenever `cgi_error_log_cycle_time` or `cgi_error_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is ”`error_log`”.

`cgi_suffix_enable yes | no` Allows or disallows any file with a .cgi extension to be executed as a CGI script. This can allow users to create CGI without having access to `/cgi-bin`, and the Sun WebServer administrator has no control over what CGI the server is executing. This is a potential security risk, so consider what potential .cgi files may be put on your system before enabling this feature. Enabling both `user_doc_enable` and `cgi_suffix_enable` is generally risky since it allows any user to execute arbitrary .cgi files through Sun WebServer. Default is “`no`”.

`cgi_user` *username*
: Sets the user name that will be used to run CGI scripts. By setting the `cgi_user`, you can control which files and services CGI scripts can affect. The `cgi_user` must have execute permission for scripts in a host's `/cgi-bin` directory. Default is "`nobody`".

`content_file` *relative_path*
: Specifies the path to the file containing the content for meta data. Path is relative to `site_path`. Default is "`content.conf`".

`content_digest yes | no`
: Allows or disallows MD5 checksum to be sent with the response entity. Default is "`no`".

`default_file` *file* [*file*]...
: Sets the name of the file Sun WebServer will look for in a directory when a URL request does not name a specific file. For example, if the URL request is for http://hostname/, Sun WebServer will look at the top directory of hostname's doc_root for a file named *default_file*.

  If `default_file` does not appear in the configuration file, then "index.html" is used. If `default_file` is " ", then no default file is used. If multiple files are specified, then the files are used in the specified order.

  If no file matching the values for default_file is found, the directory contents will be listed, subject to the value of `directory_listing`.

  The built in server-wide default is "`index.html`". The listings are in order of preference.

`directory_listing fancy | simple | off`
: Specifies how the contents of directories will be listed if no file matching `default_file` is found in the directory.

  `fancy`
  : Displays directory contents with each name as a hyperlink to the file, icons matching each file's type, and file size and date

		information. The icons used and the association of icons to file suffixes is configurable.
	`simple`	Displays only each file name as a hyperlink to the file itself.
	`off`	Disables displaying directory contents; a HTTP "404 Not Found" error is returned to the client instead.
	The built-in server-wide default is “`fancy`”.	
`doc_root` *relative_path*	Sets the top-level directory available to the host for serving WWW documents. Except for special cases (such as `/cgi-bin` and users’ personal directories), Sun WebServer will only have access to the file system at the `doc_root` and its subdirectories. This must be set for every host, but it does not have to be unique. You should create a new `doc_root` for your site. The default is "`/var/http/demo/public`". Path is relative to `site_path`.	
`error_document` <*http error code*> *url*	Allows customized error messages to be returned to the client. Any valid URL, including CGI scripts, may be returned, so you have flexibility in what information you want to give clients when an error occurs. When an HTTP error code is returned, Sun WebServer will return an HTTP "302: Document Moved" header with a Location: header indicating the file to which the error has been remapped. Most clients will automatically fetch the URL named by the Location: header. You can redirect the following HTTP error codes: ■ 400 - Bad Request (Remapped by default host only) ■ 403 - Forbidden	

- 404 - Not Found
- 412 - Precondition Failed
- 500 - Server Error
- 501 - Not Implemented
- 503 - Service Unavailable

The destination URL can be relative to the current or default host if it begins with a (/). Otherwise, an absolute URL must be specified. The following examples show a relative and absolute URL, respectively:

- error_document "503"
- "/cgi-bin/error.pl?503"
- error_document "500"
- "http://www2.A.com/mirror/"

Note - The path to which you remap 404 errors must be available in the document root. You can not redirect 404 errors to aliased directories.

`icon_add` *alt_text bitmap_URI file_type*[*file_type*]... Special file types are UP | DIR and file_type = "`..`" or "`/`"

Allows association of a file extension with an icon file. The icon will be used to represent all files with the extension in `fancy` directory listings. `icon_add` can also change a default association or assign icons to file types not covered in the default set; for example, you may want to use your own icons to represent basic types.

alt text	Specifies a string that will be used instead of an icon in text-only browsers. For example, "`GIF`".
URI path to bitmap	Specifies a URI path relative to the default host (in the `server {}` block)

or the host named by the current `url {}` block. For example, "`/icons/binary.xbm`".

file suffix(es) | content type(s) Specifies a string of one or more file extensions or content type definitions that will use the icon in "`fancy`" directory listings. For example, "jpg jpeg JPG image/jpeg" assigns the icon to all files ending in .jpg, .jpeg, and .JPG or files with a MIME type of image/jpeg.

In addition to file suffixes, you can customize the icon used for "parent directory" (`../`) and "subdirectory" by specifying "UP" or "DIR" as the *alt text*.

`icon_default` *bitmap_URI* Sets the icon used for files with extensions that do not have a defined icon type. The path to the bit map file must begin with a (/), and it is relative to the default host or the host defined by the current `url {}` block.

`log_cycle_time` *minutes* Sets the maximum age, in minutes, for request log files for the current host. If a request log is older than the number of `log_cycle_time` minutes, then a new request log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is 1440 minutes (1 day).

`log_max_files` *integer* Sets the maximum number of request log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is

incremented when a new log file is created. If the sequence goes beyond `log_max_files`, it is reset to 1 and the first request log file is overwritten. This prevents the number of log files from growing without limit. Set `log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`log_max_size` *bytes*
: Sets the maximum file size, in bytes, for request log files for the current host. If a request log exceeds `log_max_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`log_prefix` *absolute_path/prefix*
: Sets the directory and log file name prefix for request logs for a given host. The string must have an absolute path name followed by the prefix that will be used for this host's log files. Separate virtual hosts must have different prefix names so that there is no conflict in writing to the logs. Each host's log files will have a name in the form:

  `<absolute pathname>/prefix.<log_type>.<sequence>`

  For example, `/var/http/logs/http.elf.2`. Sequence is incremented and a new file created whenever `log_cycle_time` or `log_max_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is"`var/http/logs/http`".

`log_translate_ip_address yes | no`
: Determines whether a given virtual host logs its IP address or DNS name into its log files. Default is "`no`" and the IP address is logged.

`log_type none | clf | elf | celf`
: Sets the format for a host's request logs. Valid parameters are:

  `none` — Disallows logging performed for this host.

  `clf` — Logs in common log file format, a format

		widely used by servers and analysis tools.
	`elf`	Logs in extended log file format.
	`eclf`	Logs in extended common log file format, provides more fields than common log format.

If `log_type` is not "`none`", then `log_prefix` must be set. Default is “`elf`”.

`map_file` *filename*

Creates an alias to a path on the file system or a redirection to a remote URL from a URI on the host. The URI token that is redirected does not have to correspond to any actual resource. A directory alias can also specify "CGI" as a class token, and all files in the directory and its subdirectories will be treated as CGI scripts. Typically, each host has a map directive to define the location of `/cgi-bin`. An example of aliasing the `/cgi-bin` directory is `map "/cgi-bin" "/var/http/www/cgi/"` CGI If the class is "`REMOTE`" map assumes the second parameter is a URL. A URL can begin with a "`/`" to denote a URI path on the same host, or it can be a fully defined URL in the form `protocol://hostname:port/URI-path`. When there is a request for a redirected URL, Sun WebServer returns an HTTP "302 Moved Temporarily" with a Location: header containing the new URL. Most clients will automatically fetch the new URL instead of displaying the error message. For example, to redirect requests for a file to another server: `map "/file/archive.html" "http://www.A.com/archives/" "REMOTE"` If map has only one parameter, it disables any redirects or aliases for that token (for example, if you want a server-wide alias to be disabled on a particular host).

	Path is relative to site_path. Default is "`map.conf`".
`mime_default_type` *type/subtype*	Sets the MIME type that will be used for files whose extension do not match any other MIME type. The default is "`text/html`".
`mime_file` *relative_path*	Contains the default MIME types definitions used for the server host. Path is relative to site_path. Default is "`mime.types`". The order in which Sun WebServer will attempt to match a file with a MIME type is as follows: 1. Check any host-specific `mime_add` definitions. 2. Check any `mime_add` and `mime_file` definitions in the `server {}` block. 3. If none of the above produces a MIME type for the file, the `mime_default_type` is used. Entries in the mime_file have the form: `<type>/<subtype><file suffix(es)>` For example: text/html html htm The default server-wide `mime_file` is `/etc/http/mime.types`.
`publish_enable yes | no`	Allows or disallows puts and deletes. The default is "`no`".
`realm_file` *path*	Specifies path to realm file relative to `site_path`. Default is "`realms.conf`".
`se_enable yes | no`	Enables or disables the servlet engine. The default is "`no`".
`servlet_token` *URI path*	Specifies the token which characterizes the request as a servlet request. Default is "`/servlet`".
`ssi_enable yes | no`	Enables or disables the ability to use server-side include statements in .shtml files. May be set differently for each virtual host on the server. Default is "`no`".
`ssi_exec yes | no`	Enables or disables the ability to use #exec server-side include directive. #exec will execute shell commands or CGI scripts, which may pose

security risks and/or slow performance. Default is “no”.

`ssi_xbithack` *string* — Specifies SSI file suffix. Default is “`.shtml`”.

`ssi_suffix full | on | off` — Allows any file in the document root with its executable bit set to be treated as a server-parsed HTML file. `ssi_suffix` can be "`full`" or "`on`" only if `ssi_enable` is "`yes`". If "`full`", Sun WebServer treats any file with the user or group executable bit set as server-parsed HTML, and it sends the file’s Last-Modified time in the HTTP header if the user executable bit is set. If "`on`", Sun WebServer only treats files with the user’s executable bit set as server-parsed HTML. Default is “`off`”.

Note - If `cgi_suffix_enable` is also enabled, Sun WebServer will check to see if an executable file is a CGI script before parsing it for server-side includes.

`symlink_follow yes | no` — Follows or ignores symbolic links in the filesystem. Ignoring symbolic links may cause a performance loss as the filename and each directory in the path of a requested resource must be checked to make sure there are no symbolic links. Following symbolic links may be a security risk because a symbolic link can potentially point to a file that is outside of the `doc_root`. A symbolic link to a sensitive file (such as `/etc/passwd`) can only be made by someone with write access to the file, so the security risk is often small and easily managed by controlling who has access to the document root. Default is “`yes`”.

`user_doc_enable yes | no` — Allows or disallows UNIX users to create personal HTML directories in their home directories. Default is “`no`”.

See the NOTESNOTES section for information on user directories in a Solaris for ISPs environment.

`user_doc_root` *relative_path* — Sets the directory name users can create in their home directories for personal HTML files. When

a request comes in for //server/~username/, Sun WebServer will look for files in ~username/<user_doc_root>/. Enables access. Default is "public_html".

See the NOTESNOTES section for information on user directories in a Solaris for ISPs environment.

`user_doc_source` *source* — Specifies the source of user information for user document directories if `user_doc_enable` is "yes." In most cases, the only valid value is UNIXSYS, and users are defined through the operating system (for example, in `/etc/passwd` or NIS).

In Solaris for ISPs, if virtual FTP servers have been defined in Sun Directory Services, you may set this to `ISP`. User information will be taken from the directory server, and the value of a user's `ispContentDirectory` will be used.

See the NOTESNOTES section for information on user directories in a Solaris for ISPs environment.

servlet_engine{} Directives

Specifies parameters of the servlet engine

`chaining_enable yes | no` — Enables or disables servlet chaining. This enables the servlet engine to run a sequence of servlets in a specified order to fulfil one single servlet request. Host administrators can specify a chain of servlets to be executed sequentially. Default is "yes".

`dynamic_linking_enable all | local | remote | none` — Allows the specified servlets access to dynamic libraries. Default is "local".

`file_access_enable all | local | remote | none` — Allows specified servlets to have access to file resources. e.g. read/write a file on local disk. Default is "`local`".

`network_access_enable all | local | remote | none` — Allows specified servlets to have access to network resources e.g. open a socket. Default is "`local`".

`reload_enable yes | no` — Allows or disallows the servlets to reload. If the servlet classfile changes, a servlet instance (reflecting the changes) can be reloaded.

Reloading can be performed by either the server or the host administrator, depending on whether the hosts are sharing a servlet engine instance. The default is "`yes`".

`remote_enable yes | no` Enables or disables remote servlets. Enabling remote servlets allows the servlets from remote sites to be loaded by the server. The default is "`yes`".

`properties_file` *relative_path* Path to "`servlets.properties`". Each servlet engine instance can have a list of preloaded servlets that it wishes to load and initialize as soon as the server starts. This list is specified in a servlet properties file. Path can be either absolute or relative to site_path.

`servlets_path` *absolute_path* Specifies the path to all loadable local servlets. This is a colon separated list of directories and jar files where the servlet engine will look for servlets (e.g. /directory/directory:/directory/directory). Default is "`site_path/servlets`".

`se_log_enable yes | no` Enables or disables the servlet error logging. The default is "`no`".

`se_log_cycle_time` *minutes* Sets the maximum age, in minutes, for servlet error log files for the current host. If a servlet error log is older than the number of `se_log_cycle_time` minutes, then a new servlet error log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is 1440 minutes (1 day).

`se_log_max_files` *integer* Sets the maximum number of servlet error log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `se_log_max_files`, it is reset to 1 and the first servlet error log file is overwritten. This prevents the number of log files from growing without limit. Set `se_log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`se_log_max_file_size` *bytes* Sets the maximum file size, in bytes, for servlet error log files for the current host. If a servlet error log exceeds `se_log_max_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`se_log_prefix` *absolute_path/prefix* Sets the directory and log file name prefix for servlet error logs for a given host. The string must have an absolute pathname followed by the prefix that will be used for this host's log files. Separate virtual hosts must have different prefix names so that there is no conflict in writing to the logs. Each of a hosts log files will have a name in the form:

`<absolute pathname>/prefix.<sequence>`

For example, `site_path/logs/se_logs/http.2`. Sequence is incremented and a new file created whenever `se_log_cycle_time` or `se_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is "`/var/http/logs/se_logs`".

`security_access_enable` `all|local|remote|none` Allows the specified servlets to have access to security resources, e.g. classLoaders. Default is "`none`".

`system_access_enable` `all|local|remote|none` Allows or disallows the specified servlets to have access to system resources, e.g. call `System.Exec()`. Default is "`local`".

EXAMPLES

EXAMPLE 1

A sample web site configuration file with an enabled servlet engine:

```
#
# Copyright (c) 1998, by Sun Microsystems, Inc.
# All rights reserved.
#
# Sun WebServer Site Configuration

url  {
    doc_root    /var/http/sws_server/websites/www.A.com/public
    log_type    eclf
```

```
    user_doc_enable     yes

    map_file            conf/map.conf
    realm_file          conf/realms.conf
    access_file         conf/access.conf
    content_file        conf/content.conf

    # If you need more than the global mime types in
    # /etc/http/mime.types, create this file:
    #
    # mime_file         conf/mime.types

    se_enable                   yes
    servlet_token               "/servlet/"
    servlet_engine {
        properties_file        /var/http/sws_server/websites/www.A.com/conf/servlets.properties
        servlets_path          //var/http/sws_server/websites/www.A.com/servlets/
        se_log_prefix          "/var/http/logs/A.com_se_log"
        se_log_enable          yes
        se_log_max_files       7
        se_log_max_file_size   1048576
        se_log_cycle_time      1440

        reload_enable          yes
        remote_enable          yes
        chaining_enable        yes
        network_access_enable         local
        file_access_enable            local
        dynamic_linking_enable        local
        system_access_enable          local
        security_access_enable        local
    }
}
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

httpd.conf(4), **hthost**(1m)

NOTES

In a Solaris for ISPs environment, user information (including user content directories) can be accessed through LDAP and Sun Directory Services.

To use LDAP for user content directories, a virtual FTP site matching the Sun WebServer web site (virtual host) must exist in the Directory Services. Refer to the Sun™ Internet FTP Server 1.0 online help for information on creating virtual FTP sites.

If `user_doc_source` is set to "`ISP`," Sun WebServer will attempt to map `~user` URLs to the value of `ispContentDirectory` in Sun Directory Services.

The `ispContentDirectory` is relative to the `ispRootDirectory` defined in the `ispService` entry for a virtual FTP site. The file system directory must be readable and writeable by either the `uidNumber` or `gidNumber` defined in a subscriber's entry in the directory service.

NAME

map.conf – Creates an alias to a path on the file system or a redirection to a remote URL from a URI on the host.

SYNOPSIS

site_path`/conf/map.conf`

DESCRIPTION

A map directive allows you to redirect requests for a URL on a host to any other URL or to a different directory. `htmap` administers maps to establish an alias to another resource, make a resource outside of the doc_root accessible to a client, or partition the name space into various classes of resources such as CGI, imagemap, or servlet. The URL token that is redirected does not have to correspond to any actual resource.

Servlets use maps for creating servlet chains or file aliases.

A map defintion consists of the following:

- URI token
- Resource target
- Class type (optional)

Syntax

Each line in the `map.conf` file has the following format:

uri_token resource_target class_type

URI_token Specifies a URI path on the given host beginning with a slash (/).

resource_target Specifies a URI path on the given host if it begin with a slash (/), or an absolute URL to any network location if it does not begin with a (/). For either local or remote redirects, *class_type* must be REMOTE. It can also be an absolute path on the file system if it begins with a (/) and the *class_type* is ADMIN, CGI, IMAP, NULL, SERVLET, or STATS.

class_type	Specifies the class type. Valid entries are as follows:	
	ADMIN	Treats the resource target as the Sun WebServer administration directory.
	CGI	Treats the resource target as a CGI directory. All files located here will be treated as executable scripts.
	IMAP	Treats the resource target as an imagemap alias.
	NULL	Treats the resource target as a regular file alias.
	REMOTE	Treats the resource target as a remote alias.
	STATS	Treats the resource target as an interface to server statistics.
	SERVLET	Treats the resource target as a servlet or a chain of servlets. URIs which begin with the aliased token are treated as requests for servlets.

EXAMPLES

EXAMPLE 1

A directory alias can specify “CGI” as a class token, and all files in the directory and its subdirectories will be treated as CGI scripts. Typically,

host has a map directive to define the location of `/cgi-bin`. An example of creating an alias for the `/cgi-bin` is

```
map "/cgi-bin" "/var/http/www/cgi/" CGI
```

EXAMPLE 2

A directory alias can also specify “SERVLET” as a class token, and the aliased *resource_target* is treated as a servlet or a chain of servlets. An example of creating an alias for the *resource_target* is

```
map "/foo" "myFoo" SERVLET
```

where “myFoo” is a servlet defined in `servlet.properties`.

Another example of creating an alias for the *resource_target* is

```
map "/combo" "S1,S2" SERVLET
```

where ”/combo” defines a chain of servlets and “S1” and “S2” are servlets defined in `servlet.properties`.

EXAMPLE 3

If map has only one parameter, it disables any redirects or aliases for that token.

```
map   /cgi-bin    /var/http/public/x/y    CGI
map   /search     /searchServlet    SERVLET
```

(Continuation)

```
map   /home.html   /var/http/docs/index.html   NULL
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htmap(1m)

NAME

realms.conf – Defines realms of user and group information used by access control lists on a Sun™ WebServer™ web site.

SYNOPSIS

site_path`/conf/realms.conf`

DESCRIPTION

A realm defines a protection space, a domain of users and groups and their permissions. Sun WebServer access control configuration permits realms that are configured in `realms.conf` with the ACLs.

Realm files may be created by **`htrealm`**(1m)or by the Sun WebServer Administration Console.

A realm definition consists of the following:

- Realm identifier
- Source of user information: `HTPASSWD`, `ISP`, `ISPADMIN`, or `UNIXSYS`

Note - ISP or ISPADMIN realms are only valid if you are running Sun WebServer in an environment where directory service for Solaris™ for ISPs™ has been installed.

- List of realm members with permission to modify the realm itself
- Directory location of user information for HTPASSWD realms.

Syntax

The following syntax rules apply to the `realms.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks which also use matched curly braces to contain a definition.

Each realm definition is in the following form:

```
realm <identifier> {
  realm_source UNIXSYS | ISP | ISPADMIN | HTPASSWD
```

(continued)

(Continuation)

```
  [ realm_dir <data_directory> ]
  administrator {
    user <realm_user_name>[ <realm_user_name>...]
    [ group <realm_group_name>[ <realm_group_name>...] ]
  }
}
```

The syntax and definition of each directive are explained in the following *Directives* section.

Directives

The following keyword directives are valid in the `realms.conf` file:

`administrator { `*admins*` }` Defines the realm users and groups that have permission to modify realm data. The *admins* directive may have a `user` directive or optionally have a `group` directive.

`group` *group_name*`[ `*group_name*`... ]` Names groups of users that have permission to modify realm data. The *group_name* directive is an optional directive valid in the `administrator` block. Separate multiple group names with white space.

`isp_component` *component_definition* Defines the component identification, version of Solaris for ISPs, and the Administrator realm (ISPADMIN). The default value is "SUNWhttp_2.0".

`realm` *identifier* `{` *definition* `}` Defines a realm. There may be multiple realm definitions in the `realms.conf` file, as long as each has a unique *identifier*. The *identifier* directive can be any arbitrary string of alphanumeric data (no special characters). White space is allowed when enclosed in double quotes.

The *definition* consists of realm directives, and must include at least a `realm_source`.

[`realm_dir` *data_path***]** Defines a directory relative to the site path where the `users` and `groups` files for an `HTPASSWD` realm are stored. `realm_dir` is required and valid only if `realm_source` is `HTPASSWD`. It can

be either an absolute path or a path relative to realms.conf, or it can be left unspecified. The default value is `realms/`*`realmname`*`/`.

`realm_source` *source*

Defines the source of user and group information for the realm. This directive is required in every realm definition. `realm_source` may be one of the following:

`HTPASSWD`	Indicates that the user or group information is retrieved using the Sun WebServer users/group file format, and that user and group information will be maintained in the data directory named by `realm_dir`. The **`htrealm`**(1m) utility is used to create users and modify passwords.
`ISP`	Indicates that the user or group is stored in the Solaris for ISPs shared directory service. Changes to user and group information cannot be made through Sun WebServer.
`ISPADMIN`	Indicates that the principals are Administrators in the Solaris for ISPs Management Console. The `–d` flag takes the ISP-component ID

and version (for example, “SUNWfinger–1.0”).

`UNIXSYS` — Indicates that the operating system user and group definitions will be used to authenticate users in the realm. Changes to user and group information cannot be made through Sun WebServer.

`user` *user_name*`[` *user_name*`...]` — Names realm users that have permission to modify realm data. The *user_name* is a required directive valid in the `administrator` block. Separate multiple user names with white space.

EXAMPLES

EXAMPLE 1

/etc/pam.conf

```
httpd auth sufficient /usr/lib/security/pam_unix.so.1
httpd-isp auth sufficient /usr/lib/security/pam_ldap.so.1 autohost
```

This configures Sun WebServer httpd to use the UNIX PAM library for authenticating for Solaris for ISPs subscribers stored in the LDAP-based directory. This does not use stacking, but uses different service names (httpd, httpd-isp).

CODE EXAMPLE 1

Sample `realms.conf` file:

```
realm siteAdmin {
        realm_source HTPASSWD
        administrator {
                user user1
         }

}
```

```
realm SystemUsers {
        realm_source UNIXSYS

}
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htrealm(1m)

NOTES

Sun WebServer on Solaris 2.6 and greater uses a pluggable authentication module (PAM) for authenticating principals in UNIXSYS and ISP realms using `/usr/lib/security/pam_unix.so`, and `/usr/lib/security/pam_ldap.so` respectively. Refer to **pam.conf**(4) for details on how to set up PAM.

NAME

servlets.properties – Defines servlet properties file in a general Java™ properties file format.

SYNOPSIS

site_path/conf/servlets.properties

DESCRIPTION

This file contains the name of each servlet and the initialization parameters of the servlet. You can either place the properties file in the default location, or specify the name of the servlets.properties file on startup using the command `htservlet` and specifying –`i` *hostname* for the host name and –`p` *properties_file* for the name of the properties file.

The servlet.properties file may be updated using the Sun™ WebServer™ Administration Console or the **`htservlet`**(1m) utility.

You can specify the following servlet properties definitions:

- Code
- Code base
- Initial arguments

Syntax

The definitions in the `servlets.properties` file have the following format:

```
servlet.<servlet_name>.code=<servletclass>
servlet.<servlet_name>.codebase=<URL>
servlet.<servlet_name>.initArgs=<name>=<value>[,<name>=value...]
```

The following keyword directives are valid in the `servlet.properties` file:

servlet.<servlet_name>.code=<servletclass> Name of the servlet main class file.

servlet.<servlet_name>.codebase=<URL> Name of the jar file or the URL of the servlet's codebase. Used only for remote servlets.

servlet.<servlet_name>.initArgs=<name>=<value>[,<name>=value...] Optional initialization parameters passed to the servlet. Used in the format *name=value* [, *name=value*...]

EXAMPLES

EXAMPLE 1

A sample `servlets.properties` based on the default file:

```
#
# Copyright (c) 1998, by Sun Microsystems, Inc.
# All rights reserved.
```

```
#
# servlets.properties
#
#       Sample SWS Servlets Properties File
#
################################################################################


# the list of servlets to be loaded at startup time
servlets.startup=simple hello


# Date Servlet -- JSDK Sample Servlet
servlet.date.code=DateServlet

# Finger Servlet -- JSDK Sample Servlet
servlet.finger.code=FingerServlet

# Hello World  Servlet -- JSDK Sample Servlet
servlet.hello.code=HelloWorldServlet

# Snoop Servlet -- JSDK Sample Servlet
servlet.snoop.code=SnoopServlet
servlet.snoop.initArgs= port=23, host=localhost

# Simple Servlet -- JSDK Sample Servlet
servlet.simple.code=SimpleServlet
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htservlet(1m), **httpd.conf**(4)