Apache MPM Common Directives

Description:	A collection of directives that are implemented by more than one multi-processing module (MPM)
Status:	MPM

Topics

Directives

AcceptMutex Directive

Description:	Method that Apache uses to serialize multiple children accepting requests on network sockets
Syntax:	`AcceptMutex Default|`*`method`*
Default:	`AcceptMutex Default`
Context:	server config
Status:	MPM
Module:	`leader`, `perchild`, `prefork`, `threadpool`, `worker`

The `AcceptMutex` directives sets the method that Apache uses to serialize multiple children accepting requests on network sockets. Prior to Apache 2.0, the method was selectable only at compile time. The optimal method to use is highly architecture and platform dependent. For further details, see the performance tuning[1] documentation.

If this directive is set to `Default`, then the compile-time selected default will be used. Other possible methods are listed below. Note that not all methods are available on all platforms. If a method is specified which is not available, a message will be written to the error log listing the available methods.

`flock`
uses the `flock(2)` system call to lock the file defined by the `LockFile` directive.

`fcntl`
uses the `fnctl(2)` system call to lock the file defined by the `LockFile` directive.

`posixsem`
uses POSIX compatible semaphores to implement the mutex.

`pthread`
uses POSIX mutexes as implemented by the POSIX Threads (PThreads) specification.

`sysvsem`

uses SySV-style semaphores to implement the mutex.

If you want to find out the compile time chosen default for your system, you may set your `LogLevel` to `debug`. Then the default `AcceptMutex` will be written into the `ErrorLog`.

BS2000Account Directive

Description:	Define the non-privileged account on BS2000 machines
Syntax:	`BS2000Account` *`account`*
Context:	server config
Status:	MPM
Module:	`perchild`, `prefork`
Compatibility:	Only available for BS2000 machines

The `BS2000Account` directive is available for BS2000 hosts only. It must be used to define the account number for the non-privileged apache server user (which was configured using the `User` directive). This is required by the BS2000 POSIX subsystem (to change the underlying BS2000 task environment by performing a sub-LOGON) to prevent CGI scripts from accessing resources of the privileged account which started the server, usually `SYSROOT`.

Note

Only one `BS2000Account` directive can be used.

See also

- Apache EBCDIC port[2]

CoreDumpDirectory Directive

Description:	Directory where Apache attempts to switch before dumping core
Syntax:	`CoreDumpDirectory` *`directory`*
Default:	`See usage for the default setting`
Context:	server config
Status:	MPM
Module:	`beos`, `leader`, `mpm_winnt`, `perchild`, `prefork`, `threadpool`, `worker`

This controls the directory to which Apache attempts to switch before dumping core. The default is in the `ServerRoot` directory, however since this should not be writable by the user the server runs as, core dumps won't normally get written. If you want a core dump for debugging, you can use this directive to place it in a different location.

Group Directive

Description:	Group under which the server will answer requests
Syntax:	`Group` *`unix-group`*
Default:	`Group #-1`
Context:	server config

Status:	MPM
Module:	beos, leader, mpmt_os2, perchild, prefork, threadpool, worker
Compatibility:	Only valid in global server config since Apache 2.0

The Group directive sets the group under which the server will answer requests. In order to use this directive, the server must be run initially as root. If you start the server as a non-root user, it will fail to change to the specified group, and will instead continue to run as the group of the original user. *Unix-group* is one of:

A group name
Refers to the given group by name.

 # followed by a group number.
Refers to a group by its number.

Example
```
Group www-group
```

It is recommended that you set up a new group specifically for running the server. Some admins use user nobody, but this is not always possible or desirable.

Security
Don't set Group (or User) to root unless you know exactly what you are doing, and what the dangers are.

Special note: Use of this directive in <VirtualHost> is no longer supported. To configure your server for suexec[3] use SuexecUserGroup.

Note
Although the Group directive is present in the beos and mpmt_os2 MPMs, it is actually a no-op there and only exists for compatibility reasons.

Listen Directive

Description:	IP addresses and ports that the server listens to
Syntax:	Listen [*IP-address*:]*portnumber*
Context:	server config
Status:	MPM
Module:	beos, leader, mpm_netware, mpm_winnt, mpmt_os2, perchild, prefork, threadpool, worker
Compatibility:	Required directive since Apache 2.0

The Listen directive instructs Apache to listen to only specific IP addresses or ports; by default it responds to requests on all IP interfaces. Listen is now a required directive. If it is not in the config file, the server will fail to start. This is a change from previous versions of Apache.

The Listen directive tells the server to accept incoming requests on the specified port or address-and-port combination. If only a port number is specified, the server listens to the given port on all interfaces. If an IP address is given as well as a port, the server will listen on the given port and

interface.

Multiple `Listen` directives may be used to specify a number of addresses and ports to listen to. The server will respond to requests from any of the listed addresses and ports.

For example, to make the server accept connections on both port 80 and port 8000, use:

```
Listen 80
Listen 8000
```

To make the server accept connections on two specified interfaces and port numbers, use

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6 addresses must be surrounded in square brackets, as in the following example:

```
Listen [fe80::a00:20ff:fea7:ccea]:80
```

See also

- DNS Issues[6]
- Setting which addresses and ports Apache uses[7]

ListenBackLog Directive

Description:	Maximum length of the queue of pending connections
Syntax:	`ListenBacklog` *`backlog`*
Default:	`ListenBacklog 511`
Context:	server config
Status:	MPM
Module:	`beos`, `leader`, `mpm_netware`, `mpm_winnt`, `mpmt_os2`, `perchild`, `prefork`, `threadpool`, `worker`

The maximum length of the queue of pending connections. Generally no tuning is needed or desired, however on some systems it is desirable to increase this when under a TCP SYN flood attack. See the backlog parameter to the `listen(2)` system call.

This will often be limited to a smaller number by the operating system. This varies from OS to OS. Also note that many OSes do not use exactly what is specified as the backlog, but use a number based on (but normally larger than) what is set.

LockFile Directive

Description:	Location of the accept serialization lock file
Syntax:	`LockFile` *`filename`*
Default:	`LockFile logs/accept.lock`
Context:	server config
Status:	MPM

Module:	leader, perchild, prefork, threadpool, worker

The LockFile directive sets the path to the lockfile used when Apache is used with an AcceptMutex value of either fcntl or flock. This directive should normally be left at its default value. The main reason for changing it is if the logs directory is NFS mounted, since **the lockfile must be stored on a local disk**. The PID of the main server process is automatically appended to the filename.

> **Security:**
>
> It is best to *avoid* putting this file in a world writable directory such as /var/tmp because someone could create a denial of service attack and prevent the server from starting by creating a lockfile with the same name as the one the server will try to create.

See also

- AcceptMutex

MaxClients Directive

Description:	Maximum number of child processes that will be created to serve requests
Syntax:	MaxClients *number*
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	beos, leader, prefork, threadpool, worker

The MaxClients directive sets the limit on the number of simultaneous requests that will be served. Any connection attempts over the MaxClients limit will normally be queued, up to a number based on the ListenBacklog directive. Once a child process is freed at the end of a different request, the connection will then be serviced.

For non-threaded servers (*i.e.*, prefork), MaxClients translates into the maximum number of child processes that will be launched to serve requests. The default value is 256; to increase it, you must also raise ServerLimit.

For threaded and hybrid servers (*e.g.* beos or worker) MaxClients restricts the total number of threads that will be available to serve clients. The default value for beos is 50. For hybrid MPMs the default value is 16 (ServerLimit) multiplied by the value of 25 (ThreadsPerChild). Therefore, to increase MaxClients to a value that requires more than 16 processes, you must also raise ServerLimit.

MaxMemFree Directive

Description:	Maximum amount of memory that the main allocator is allowed to hold without calling free()
Syntax:	MaxMemFree *KBytes*
Default:	MaxMemFree 0
Context:	server config
Status:	MPM
Module:	beos, leader, mpm_netware, prefork, threadpool, worker

The MaxMemFree directive sets the maximum number of free Kbytes that the main allocator is allowed to hold without calling free(). When not set, or when set to zero, the threshold will be set to unlimited.

MaxRequestsPerChild Directive

Description:	Limit on the number of requests that an individual child server will handle during its life
Syntax:	MaxRequestsPerChild *number*
Default:	MaxRequestsPerChild 10000
Context:	server config
Status:	MPM
Module:	leader, mpm_netware, mpm_winnt, mpmt_os2, perchild, prefork, threadpool, worker

The MaxRequestsPerChild directive sets the limit on the number of requests that an individual child server process will handle. After MaxRequestsPerChild requests, the child process will die. If MaxRequestsPerChild is 0, then the process will never expire.

Different default values:

The default value for mpm_netware and mpm_winnt is 0.

Setting MaxRequestsPerChild to a non-zero limit has two beneficial effects:

- it limits the amount of memory that process can consume by (accidental) memory leakage;
- by giving processes a finite lifetime, it helps reduce the number of processes when the server load reduces.

Note:

For KeepAlive requests, only the first request is counted towards this limit. In effect, it changes the behavior to limit the number of *connections* per child.

MaxSpareThreads Directive

Description:	Maximum number of idle threads
Syntax:	MaxSpareThreads *number*
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	beos, leader, mpm_netware, mpmt_os2, perchild, threadpool, worker

Maximum number of idle threads. Different MPMs deal with this directive differently.

For perchild the default is MaxSpareThreads 10. This MPM monitors the number of idle threads on a per-child basis. If there are too many idle threads in that child, the server will begin to kill threads within that child.

For worker, leader and threadpool the default is MaxSpareThreads 250. These MPMs deal with idle threads on a server-wide basis. If there are too many idle threads in the server then child processes are killed until the number of idle threads is less than this number.

For mpm_netware the default is MaxSpareThreads 100. Since this MPM runs a single-process, the spare thread count is also server-wide.

beos and mpmt_os2 work similar to mpm_netware. The default for beos is MaxSpareThreads 50. For mpmt_os2 the default value is 10.

Restrictions

The range of the MaxSpareThreads value is restricted. Apache will correct the given value automatically according to the following rules:

- perchild requires MaxSpareThreads to be less or equal than ThreadLimit.
- mpm_netware wants the value to be greater than MinSpareThreads.
- For leader, threadpool and worker the value must be greater or equal than the sum of MinSpareThreads and ThreadsPerChild.

See also

- MinSpareThreads
- StartServers

MinSpareThreads Directive

Description:	Minimum number of idle threads available to handle request spikes
Syntax:	MinSpareThreads *number*
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	beos, leader, mpm_netware, mpmt_os2, perchild, threadpool, worker

Minimum number of idle threads to handle request spikes. Different MPMs deal with this directive differently.

perchild uses a default of MinSpareThreads 5 and monitors the number of idle threads on a per-child basis. If there aren't enough idle threads in that child, the server will begin to create new threads within that child. Thus, if you set NumServers to 10 and a MinSpareThreads value of 5, you'll have at least 50 idle threads on your system.

worker, leader and threadpool use a default of MinSpareThreads 75 and deal with idle threads on a server-wide basis. If there aren't enough idle threads in the server then child processes are created until the number of idle threads is greater than number.

mpm_netware uses a default of MinSpareThreads 10 and, since it is a single-process MPM, tracks this on a server-wide bases.

beos and mpmt_os2 work similar to mpm_netware. The default for beos is MinSpareThreads 1. For mpmt_os2 the default value is 5.

See also

- MaxSpareThreads
- StartServers

PidFile Directive

Description:	File where the server records the process ID of the daemon
Syntax:	`PidFile` *`filename`*
Default:	`PidFile logs/httpd.pid`
Context:	server config
Status:	MPM
Module:	beos, leader, mpm_winnt, mpmt_os2, perchild, prefork, threadpool, worker

The PidFile directive sets the file to which the server records the process id of the daemon. If the filename is not absolute then it is assumed to be relative to the ServerRoot.

Example

```
PidFile /var/run/apache.pid
```

It is often useful to be able to send the server a signal, so that it closes and then re-opens its ErrorLog and TransferLog, and re-reads its configuration files. This is done by sending a SIGHUP (kill -1) signal to the process id listed in the PidFile.

The PidFile is subject to the same warnings about log file placement and security[4].

Note

As of Apache 2 it is recommended to use only the apachectl[5] script for (re-)starting or stopping the server.

ScoreBoardFile Directive

Description:	Location of the file used to store coordination data for the child processes
Syntax:	`ScoreBoardFile` *`file-path`*
Default:	`ScoreBoardFile logs/apache_status`
Context:	server config
Status:	MPM
Module:	beos, leader, mpm_winnt, perchild, prefork, threadpool, worker

Apache uses a scoreboard to communicate between its parent and child processes. Some architectures require a file to facilitate this communication. If the file is left unspecified, Apache first attempts to create the scoreboard entirely in memory (using anonymous shared memory) and, failing that, will attempt to create the file on disk (using file-based shared memory). Specifying this directive causes Apache to always create the file on the disk.

Example

```
ScoreBoardFile /var/run/apache_status
```

File-based shared memory is useful for third-party applications that require direct access to the scoreboard.

If you use a `ScoreBoardFile` then you may see improved speed by placing it on a RAM disk. But be careful that you heed the same warnings about log file placement and security[8].

See also

- Stopping and Restarting Apache[9]

SendBufferSize Directive

Description:	TCP buffer size
Syntax:	`SendBufferSize` *`bytes`*
Default:	`SendBufferSize 0`
Context:	server config
Status:	MPM
Module:	`beos`, `leader`, `mpm_netware`, `mpm_winnt`, `mpmt_os2`, `perchild`, `prefork`, `threadpool`, `worker`

The server will set the TCP buffer size to the number of bytes specified. Very useful to increase past standard OS defaults on high speed high latency (*i.e.*, 100ms or so, such as transcontinental fast pipes).

If set to the value of `0`, the server will use the OS deault.

ServerLimit Directive

Description:	Upper limit on configurable number of processes
Syntax:	`ServerLimit` *`number`*
Default:	`See usage for details`
Context:	server config
Status:	MPM
Module:	`leader`, `perchild`, `prefork`, `threadpool`, `worker`

For the `prefork` MPM, this directive sets the maximum configured value for `MaxClients` for the lifetime of the Apache process. For the worker MPM, this directive in combination with `ThreadLimit` sets the maximum configured value for `MaxClients` for the lifetime of the Apache process. Any attempts to change this directive during a restart will be ignored, but `MaxClients` can be modified during a restart.

Special care must be taken when using this directive. If `ServerLimit` is set to a value much higher than necessary, extra, unused shared memory will be allocated. If both `ServerLimit` and `MaxClients` are set to values higher than the system can handle, Apache may not start or the system may become unstable.

With the `prefork` MPM, use this directive only if you need to set `MaxClients` higher than 256 (default). Do not set the value of this directive any higher than what you might want to set

`MaxClients` to.

With `worker`, `leader` and `threadpool` use this directive only if your `MaxClients` and `ThreadsPerChild` settings require more than 16 server processes (default). Do not set the value of this directive any higher than the number of server processes required by what you may want for `MaxClients` and `ThreadsPerChild`.

With the `perchild` MPM, use this directive only if you need to set `NumServers` higher than 8 (default).

> **Note**
>
> There is a hard limit of `ServerLimit 20000` compiled into the server. This is intended to avoid nasty effects caused by typos.

See also

- Stopping and Restarting Apache[9]

StartServers Directive

Description:	Number of child server processes created at startup
Syntax:	`StartServers` *`number`*
Default:	`See usage for details`
Context:	server config
Status:	MPM
Module:	`leader`, `mpmt_os2`, `prefork`, `threadpool`, `worker`

The `StartServers` directive sets the number of child server processes created on startup. As the number of processes is dynamically controlled depending on the load, there is usually little reason to adjust this parameter.

The default value differs from MPM to MPM. For `leader`, `threadpool` and `worker` the default is `StartServers 3`. For `prefork` defaults to `5` and for `mpmt_os2` to `2`.

StartThreads Directive

Description:	Number of threads created on startup
Syntax:	`StartThreads` *`number`*
Default:	`See usage for details`
Context:	server config
Status:	MPM
Module:	`beos`, `mpm_netware`, `perchild`

Number of threads created on startup. As the number of threads is dynamically controlled depending on the load, there is usually little reason to adjust this parameter.

For `perchild` the default is `StartThreads 5` and this directive tracks the number of threads per process at startup.

For `mpm_netware` the default is `StartThreads 50` and, since there is only a single process, this is

the total number of threads created at startup to serve requests.

For beos the default is StartThreads 10. It also reflects the total number of threads created at startup to serve requests.

ThreadLimit Directive

Description:	Sets the upper limit on the configurable number of threads per child process
Syntax:	ThreadLimit *number*
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	leader, mpm_winnt, perchild, threadpool, worker
Compatibility:	Available for mpm_winnt in Apache 2.0.41 and later

This directive sets the maximum configured value for ThreadsPerChild for the lifetime of the Apache process. Any attempts to change this directive during a restart will be ignored, but ThreadsPerChild can be modified during a restart up to the value of this directive.

Special care must be taken when using this directive. If ThreadLimit is set to a value much higher than ThreadsPerChild, extra unused shared memory will be allocated. If both ThreadLimit and ThreadsPerChild are set to values higher than the system can handle, Apache may not start or the system may become unstable. Do not set the value of this directive any higher than your greatest predicted setting of ThreadsPerChild for the current run of Apache.

The default value for ThreadLimit is 1920 when used with mpm_winnt and 64 when used with the others.

> **Note**
>
> There is a hard limit of ThreadLimit 20000 (or ThreadLimit 15000 with mpm_winnt) compiled into the server. This is intended to avoid nasty effects caused by typos.

ThreadsPerChild Directive

Description:	Number of threads created by each child process
Syntax:	ThreadsPerChild *number*
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	leader, mpm_winnt, threadpool, worker

This directive sets the number of threads created by each child process. The child creates these threads at startup and never creates more. If using an MPM like mpm_winnt, where there is only one child process, this number should be high enough to handle the entire load of the server. If using an MPM like worker, where there are multiple child processes, the *total* number of threads should be high enough to handle the common load on the server.

The default value for ThreadsPerChild is 64 when used with mpm_winnt and 25 when used with the others.

User Directive

Description:	The userid under which the server will answer requests
Syntax:	`User` *`unix-userid`*
Default:	`User #-1`
Context:	server config
Status:	MPM
Module:	`leader`, `perchild`, `prefork`, `threadpool`, `worker`
Compatibility:	Only valid in global server config since Apache 2.0

The `User` directive sets the user ID as which the server will answer requests. In order to use this directive, the server must be run initially as `root`. If you start the server as a non-root user, it will fail to change to the lesser privileged user, and will instead continue to run as that original user. If you do start the server as `root`, then it is normal for the parent process to remain running as root. *Unix-userid* is one of:

A username
: Refers to the given user by name.

 # followed by a user number.
: Refers to a user by its number.

The user should have no privileges that result in it being able to access files that are not intended to be visible to the outside world, and similarly, the user should not be able to execute code that is not meant for HTTP requests. It is recommended that you set up a new user and group specifically for running the server. Some admins use user `nobody`, but this is not always desirable, since the `nobody` user can have other uses on the system.

> **Security**
>
> Don't set `User` (or `Group`) to `root` unless you know exactly what you are doing, and what the dangers are.

With the `perchild` MPM, which is intended to server virtual hosts run under different user IDs, the `User` directive defines the user ID for the main server and the fallback for `<VirtualHost>` sections without an `AssignUserID` directive.

Special note: Use of this directive in `<VirtualHost>` is no longer supported. To configure your server for suexec[10] use `SuexecUserGroup`.

> **Note**
>
> Although the `User` directive is present in the `beos` and `mpmt_os2` MPMs, it is actually a no-op there and only exists for compatibility reasons.

URI References

[1] http://httpd.apache.org/docs-2.1/misc/perf-tuning.html
[2] http://httpd.apache.org/docs-2.1/platform/ebcdic.html
[3] http://httpd.apache.org/docs-2.1/mod/mod_suexec.html

[4] http://httpd.apache.org/docs-2.1/misc/security_tips.html#serverroot
[5] http://httpd.apache.org/docs-2.1/programs/apachectl.html
[6] http://httpd.apache.org/docs-2.1/dns-caveats.html
[7] http://httpd.apache.org/docs-2.1/bind.html
[8] http://httpd.apache.org/docs-2.1/misc/security_tips.html
[9] http://httpd.apache.org/docs-2.1/stopping.html
[10] http://httpd.apache.org/docs-2.1/suexec.html