Apache Module mod_proxy

Description:	HTTP/1.1 proxy/gateway server
Status:	Extension
Module Identifier:	proxy_module
Source File:	mod_proxy.c

Summary

Warning

This document has been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be inaccurate, please use it with care.

This module implements a proxy/gateway for Apache. It implements proxying capability for `FTP`, `CONNECT` (for SSL), `HTTP/0.9`, `HTTP/1.0`, and `HTTP/1.1`. The module can be configured to connect to other proxy modules for these and other protocols.

This module was experimental in Apache 1.1.x. Improvements and bugfixes were made in Apache v1.2.x and Apache v1.3.x, then the module underwent a major overhaul for Apache v2.0. The protocol support was upgraded to HTTP/1.1, and filter support was enabled.

Please note that the **caching** function present in mod_proxy up to Apache v1.3.x has been **removed** from mod_proxy and will be incorporated into a new module, mod_cache. In other words: the Apache 2.0.x-Proxy doesn't cache at all - all caching functionality has been moved into mod_cache, which is capable of caching any content, not only content from proxy.

If you need to use SSL when contacting remote servers, have a look at the `SSLProxy*` directives in `mod_ssl`.

Do not enable proxying with `ProxyRequests` until you have secured your server. Open proxy servers are dangerous both to your network and to the Internet at large.

Topics

Directives

Common configuration topics

- Forward and Reverse Proxies
- Controlling access to your proxy

- Why doesn't file type *xxx* download via FTP?
- How can I force an FTP ASCII download of File *xxx*?
- How can I access FTP files outside of my home directory?
- How can I hide the FTP cleartext password in my browser's URL line?
- Why does Apache start more slowly when using the proxy module?
- What other functions are useful for an intranet proxy server?
- How can I make the proxy talk HTTP/1.0 and disable keepalives?

Forward and Reverse Proxies

Apache can be configured in both a *forward* and *reverse* proxy configuration.

A *forward proxy* is an intermediate system that enables a browser to connect to a remote network to which it normally does not have access. A forward proxy can also be used to cache data, reducing load on the networks between the forward proxy and the remote webserver.

Apache's mod_proxy can be figured to behave like a forward proxy using the `ProxyRemote` directive. In addition, caching of data can be achieved by configuring Apache `mod_cache`. Other dedicated forward proxy packages include Squid[1].

A *reverse proxy* is a webserver system that is capable of serving webpages sourced from other webservers - in addition to webpages on disk or generated dynamically by CGI - making these pages look like they originated at the reverse proxy.

When configured with the mod_cache module the reverse proxy can act as a cache for slower backend webservers. The reverse proxy can also enable advanced URL strategies and management techniques, allowing webpages served using different webserver systems or architectures to coexist inside the same URL space. Reverse proxy systems are also ideal for implementing centralised logging websites with many or diverse website backends. Complex multi-tier webserver systems can be constructed using an Apache mod_proxy frontend and any number of backend webservers.

The reverse proxy is configured using the `ProxyPass` and `ProxyPassReverse` directives. Caching can be enabled using mod_cache as with the forward proxy.

Controlling access to your proxy

You can control who can access your proxy via the `<Proxy>` control block using the following example:

```
<Proxy *>
Order Deny,Allow
Deny from all
Allow from 192.168.0
</Proxy>
```

When configuring a reverse proxy, access control takes on the attributes of the normal server `<directory>` configuration.

Why doesn't file type *xxx* download via FTP?

You probably don't have that particular file type defined as *application/octet-stream* in your proxy's mime.types configuration file. A useful line can be

```
```

```
application/octet-stream bin dms lha lzh exe class tgz taz
```

How can I force an FTP ASCII download of File *xxx*?

In the rare situation where you must download a specific file using the FTP **ASCII** transfer method (while the default transfer is in **binary** mode), you can override mod_proxy's default by suffixing the request with `;type=a` to force an ASCII transfer. (FTP Directory listings are always executed in ASCII mode, however.)

How can I access FTP files outside of my home directory?

An FTP URI is interpreted relative to the home directory of the user who is logging in. Alas, to reach higher directory levels you cannot use /../, as the dots are interpreted by the browser and not actually sent to the FTP server. To address this problem, the so called "Squid %2f hack" was implemented in the Apache FTP proxy; it is is a solution which is also used by other popular proxy servers like the Squid Proxy Cache [2]. By prepending /%2f to the path of your request, you can make such a proxy change the FTP starting directory to / (instead of the home directory).

Example: To retrieve the file `/etc/motd`, you would use the URL

```
ftp://user@host/%2f/etc/motd
```

How can I hide the FTP cleartext password in my browser's URL line?

To log in to an FTP server by username and password, Apache uses different strategies. In absense of a user name and password in the URL altogether, Apache sends an anomymous login to the FTP server, i.e.,

```
user: anonymous
password: apache_proxy@
```

This works for all popular FTP servers which are configured for anonymous access.

For a personal login with a specific username, you can embed the user name into the URL, like in: `ftp://username@host/myfile`. If the FTP server asks for a password when given this username (which it should), then Apache will reply with a [401 Authorization required] response, which causes the Browser to pop up the username/password dialog. Upon entering the password, the connection attempt is retried, and if successful, the requested resource is presented. The advantage of this procedure is that your browser does not display the password in cleartext (which it would if you had used `ftp://username:password@host/myfile` in the first place).

> **Note**
>
> The password which is transmitted in such a way is not encrypted on its way. It travels between your browser and the Apache proxy server in a base64-encoded cleartext string, and between the Apache proxy and the FTP server as plaintext. You should therefore think twice before accessing your FTP server via HTTP (or before accessing your personal files via FTP at all!) When using unsecure channels, an eavesdropper might intercept your password on its way.

Why does Apache start more slowly when using the proxy module?

If you're using the `ProxyBlock` directive, hostnames' IP addresses are looked up and cached during startup for later match test. This may take a few seconds (or more) depending on the speed with which the hostname lookups occur.

What other functions are useful for an intranet proxy server?

An Apache proxy server situated in an intranet needs to forward external requests through the company's firewall. However, when it has to access resources within the intranet, it can bypass the firewall when accessing hosts. The `NoProxy` directive is useful for specifying which hosts belong to the intranet and should be accessed directly.

Users within an intranet tend to omit the local domain name from their WWW requests, thus requesting "http://somehost/" instead of "http://somehost.my.dom.ain/". Some commercial proxy servers let them get away with this and simply serve the request, implying a configured local domain. When the `ProxyDomain` directive is used and the server is configured for proxy service, Apache can return a redirect response and send the client to the correct, fully qualified, server address. This is the preferred method since the user's bookmark files will then contain fully qualified hosts.

How can I make the proxy talk HTTP/1.0 and disable keepalives?

For circumstances where you have a application server which doesn't implement keepalives or HTTP/1.1 properly, there are 2 environment variables which when set send a HTTP/1.0 with no keepalive. These are set via the `SetEnv` directive.

These are the 'force-proxy-request-1.0' and 'proxy-nokeepalive' notes.

```
<location /buggyappserver/ >
ProxyPass http://buggyappserver:7001/foo/
SetEnv force-proxy-request-1.0 1
SetEnv proxy-nokeepalive 1
</location>
```

AllowCONNECT Directive

Description:	Ports that are allowed to CONNECT through the proxy
Syntax:	`AllowCONNECT` *`port`* `[`*`port`*`] ...`
Default:	`AllowCONNECT 443 563`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `AllowCONNECT` directive specifies a list of port numbers to which the proxy `CONNECT` method may connect. Today's browsers use this method when a *https* connection is requested and proxy tunneling over *http* is in effect.
By default, only the default https port (443) and the default snews port (563) are enabled. Use the `AllowCONNECT` directive to overrride this default and allow connections to the listed ports only.

NoProxy Directive

Description:	Hosts, domains, or networks that will be connected to directly
Syntax:	`NoProxy` *`host`* `[`*`host`*`] ...`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This directive is only useful for Apache proxy servers within intranets. The `NoProxy` directive specifies a list of subnets, IP addresses, hosts and/or domains, separated by spaces. A request to a host which matches one or more of these is always served directly, without forwarding to the configured `ProxyRemote` proxy server(s).

Example

```
ProxyRemote * http://firewall.mycompany.com:81
NoProxy .mycompany.com 192.168.112.0/21
```

The *host* arguments to the NoProxy directive are one of the following type list:

Domain

A *Domain* is a partially qualified DNS domain name, preceded by a period. It represents a list of hosts which logically belong to the same DNS domain or zone (*i.e.*, the suffixes of the hostnames are all ending in *Domain*).
Examples: `.com` `.apache.org.`
To distinguish *Domain*s from *Hostname*s (both syntactically and semantically; a DNS domain can have a DNS A record, too!), *Domain*s are always written with a leading period.
Note: Domain name comparisons are done without regard to the case, and *Domain*s are always assumed to be anchored in the root of the DNS tree, therefore two domains `.MyDomain.com` and `.mydomain.com.` (note the trailing period) are considered equal. Since a domain comparison does not involve a DNS lookup, it is much more efficient than subnet comparison.

SubNet

A *SubNet* is a partially qualified internet address in numeric (dotted quad) form, optionally followed by a slash and the netmask, specified as the number of significant bits in the *SubNet*. It is used to represent a subnet of hosts which can be reached over a common network interface. In the absence of the explicit net mask it is assumed that omitted (or zero valued) trailing digits specify the mask. (In this case, the netmask can only be multiples of 8 bits wide.)
Examples:

`192.168` or `192.168.0.0`
: the subnet 192.168.0.0 with an implied netmask of 16 valid bits (sometimes used in the netmask form `255.255.0.0`)

`192.168.112.0/21`
: the subnet `192.168.112.0/21` with a netmask of 21 valid bits (also used in the form 255.255.248.0)

As a degenerate case, a *SubNet* with 32 valid bits is the equivalent to an *IPAddr*, while a *SubNet* with zero valid bits (*e.g.*, 0.0.0.0/0) is the same as the constant *_Default_*, matching any IP address.

IPAddr

A *IPAddr* represents a fully qualified internet address in numeric (dotted quad) form. Usually, this address represents a host, but there need not necessarily be a DNS domain name connected with the address.
Example: 192.168.123.7
Note: An *IPAddr* does not need to be resolved by the DNS system, so it can result in more effective apache performance.

Hostname

A *Hostname* is a fully qualified DNS domain name which can be resolved to one or more *IPAddr*s via the DNS domain name service. It represents a logical host (in contrast to *Domain*s, see above) and must be resolvable to at least one *IPAddr* (or often to a list of hosts with different *IPAddr*'s).
Examples: `prep.ai.mit.edu` `www.apache.org.`

Note: In many situations, it is more effective to specify an *IPAddr* in place of a *Hostname* since a DNS lookup can be avoided. Name resolution in Apache can take a remarkable deal of time when the connection to the name server uses a slow PPP link.
Note: *Hostname* comparisons are done without regard to the case, and *Hostname*s are always assumed to be anchored in the root of the DNS tree, therefore two hosts `WWW.MyDomain.com` and `www.mydomain.com.` (note the trailing period) are considered equal.

See also

- DNS Issues[3]

<Proxy> Directive

Description:	Container for directives applied to proxied resources
Syntax:	`<Proxy wildcard-url> ...</Proxy>`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

Directives placed in `<Proxy>` sections apply only to matching proxied content. Shell-style wildcards are allowed.

For example, the following will allow only hosts in `yournetwork.example.com` to access content via your proxy server:

```
<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from yournetwork.example.com
</Proxy>
```

The following example will process all files in the `foo` directory of `example.com` through the `INCLUDES` filter when they are sent through the proxy server:

```
<Proxy http://example.com/foo/*>
  SetOutputFilter INCLUDES
</Proxy>
```

ProxyBlock Directive

Description:	Words, hosts, or domains that are banned from being proxied
Syntax:	`ProxyBlock *|word|host|domain [word|host|domain] ...`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `ProxyBlock` directive specifies a list of words, hosts and/or domains, separated by spaces. HTTP, HTTPS, and FTP document requests to sites whose names contain matched words, hosts or domains are *blocked* by the proxy server. The proxy module will also attempt to determine IP addresses of list items which may be hostnames during startup, and cache them for match test as well. Example:

```
ProxyBlock joes-garage.com some-host.co.uk rocky.wotsamattau.edu
```

'rocky.wotsamattau.edu' would also be matched if referenced by IP address.

Note that 'wotsamattau' would also be sufficient to match 'wotsamattau.edu'.

Note also that

```
ProxyBlock *
```

blocks connections to all sites.

ProxyDomain Directive

Description:	Default domain name for proxied requests
Syntax:	`ProxyDomain` *`Domain`*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This directive is only useful for Apache proxy servers within intranets. The `ProxyDomain` directive specifies the default domain which the apache proxy server will belong to. If a request to a host without a domain name is encountered, a redirection response to the same host with the configured *Domain* appended will be generated.

Example

```
ProxyRemote * http://firewall.mycompany.com:81
NoProxy .mycompany.com 192.168.112.0/21
ProxyDomain .mycompany.com
```

ProxyErrorOverride Directive

Description:	Override error pages for proxied content
Syntax:	`ProxyErrorOverride On|Off`
Default:	`ProxyErrorOverride Off`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in version 2.0 and later

This directive is useful for reverse-proxy setups, where you want to have a common look and feel on the error pages seen by the end user. This also allows for included files (via mod_include's SSI) to get the error code and act accordingly (default behavior would display the error page of the proxied server, turning this on shows the SSI Error message).

ProxyIOBufferSize Directive

Description:	IO buffer size for outgoing HTTP and FTP connections
Syntax:	`ProxyIOBufferSize` *bytes*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

<ProxyMatch> Directive

Description:	Container for directives applied to regular-expression-matched proxied resources
Syntax:	`<Proxy` *regex*`> ...</Proxy>`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `<ProxyMatch>` directive is identical to the `<Proxy>` directive, except it matches URLs using regular expressions.

ProxyMaxForwards Directive

Description:	Maximium number of proxies that a request can be forwarded through
Syntax:	`ProxyMaxForwards` *number*
Default:	`ProxyMaxForwards 10`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in Apache 2.0 and later

The `ProxyMaxForwards` directive specifies the maximum number of proxies through which a request may pass. This is set to prevent infinite proxy loops, or a DoS attack.

Example

```
ProxyMaxForwards 10
```

ProxyPass Directive

Description:	Maps remote servers into the local server URL-space
Syntax:	`ProxyPass [`*path*`] !|`*url*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. *path* is the name of a local virtual path; *url* is a partial URL for the remote server and cannot include a

query string.

Suppose the local server has address `http://wibble.org/`; then

```
ProxyPass /mirror/foo/ http://foo.com/
```

will cause a local request for the <`http://wibble.org/mirror/foo/bar`> to be internally converted into a proxy request to <`http://foo.com/bar`>.

The ! directive is useful in situations where you don't want to reverse-proxy a subdirectory. eg.

```
ProxyPass /mirror/foo/i !
ProxyPass /mirror/foo http://foo.com
```

will proxy all requests to /mirror/foo to foo.com EXCEPT requests made to /mirror/foo/i

> NB: order is important. you need to put the exclusions BEFORE the general proxypass directive

When used inside a `<Location>` section, the first argument is ommitted and the local directory is obtained from the `<Location>`.

If you require a more flexible reverse-proxy configuration, see the `RewriteRule` directive with the `[P]` flag.

ProxyPassReverse Directive

Description:	Adjusts the URL in HTTP response headers sent from a reverse proxied server
Syntax:	`ProxyPassReverse [`*`path`*`] `*`url`*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This directive lets Apache adjust the URL in the `Location`, `Content-Location` and `URI` headers on HTTP redirect responses. This is essential when Apache is used as a reverse proxy to avoid by-passing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy.

path is the name of a local virtual path.
url is a partial URL for the remote server - the same way they are used for the `ProxyPass` directive.

Example:
Suppose the local server has address `http://wibble.org/`; then

```
ProxyPass /mirror/foo/ http://foo.com/
ProxyPassReverse /mirror/foo/ http://foo.com/
```

will not only cause a local request for the <`http://wibble.org/mirror/foo/bar`> to be internally converted into a proxy request to <`http://foo.com/bar`> (the functionality `ProxyPass` provides here). It also takes care of redirects the server foo.com sends: when `http://foo.com/bar` is redirected by him to `http://foo.com/quux` Apache adjusts this to `http://wibble.org/mirror/foo/quux` before forwarding the HTTP redirect response to the

client.

Note that this `ProxyPassReverse` directive can also be used in conjunction with the proxy pass-through feature ("`RewriteRule ... [P]`") from `mod_rewrite` because its doesn't depend on a corresponding `ProxyPass` directive.

When used inside a `<Location>` section, the first argument is ommitted and the local directory is obtained from the `<Location>`.

ProxyPreserveHost Directive

Description:	Use incoming Host HTTP request header for proxy request
Syntax:	`ProxyPreserveHost on|off`
Default:	`ProxyPreserveHost Off`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in Apache 2.0.31 and later.

When enabled, this option will pass the Host: line from the incoming request to the proxied host, instead of the hostname specified in the proxypass line.

This option should normally be turned 'off'.

ProxyReceiveBufferSize Directive

Description:	Network buffer size for outgoing HTTP and FTP connections
Syntax:	`ProxyReceiveBufferSize` *`bytes`*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `ProxyReceiveBufferSize` directive specifies an explicit network buffer size for outgoing HTTP and FTP connections, for increased throughput. It has to be greater than 512 or set to 0 to indicate that the system's default buffer size should be used.

Example

```
ProxyReceiveBufferSize 2048
```

ProxyRemote Directive

Description:	Remote proxy used to handle certain requests
Syntax:	`ProxyRemote` *`match remote-server`*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This defines remote proxies to this proxy. *match* is either the name of a URL-scheme that the remote server supports, or a partial URL for which the remote server should be used, or '*' to indicate the server should be contacted for all requests. *remote-server* is a partial URL for the remote server. Syntax:

```
remote-server = protocol://hostname[:port]
```

protocol is the protocol that should be used to communicate with the remote server; only "http" is supported by this module.

Example:

```
ProxyRemote http://goodguys.com/ http://mirrorguys.com:8000
ProxyRemote * http://cleversite.com
ProxyRemote ftp http://ftpproxy.mydomain.com:8080
```

In the last example, the proxy will forward FTP requests, encapsulated as yet another HTTP proxy request, to another proxy which can handle them.

This option also supports reverse proxy configuration - a backend webserver can be embedded within a virtualhost URL space even if that server is hidden by another forward proxy.

ProxyRemoteMatch Directive

Description:	Remote proxy used to handle requests matched by regular expressions
Syntax:	`ProxyRemote` *`regex remote-server`*
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `ProxyRemoteMatch` is identical to the `ProxyRemote` directive, except the first argument is a regular expression match against the requested URL.

ProxyRequests Directive

Description:	Enables forward (standard) proxy requests
Syntax:	`ProxyRequests on|off`
Default:	`ProxyRequests Off`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This allows or prevents Apache from functioning as a forward proxy server. (Setting ProxyRequests to 'off' does not disable use of the `ProxyPass` directive.)

In a typical reverse proxy configuration, this option should be set to 'off'.

Do not enable proxying with `ProxyRequests` until you have secured your server. Open proxy servers are dangerous both to your network and to the Internet at large.

ProxyTimeout Directive

Description:	Network timeout for proxied requests
Syntax:	`ProxyTimeout` *`seconds`*
Default:	`ProxyTimeout 300`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in Apache 2.0.31 and later

This directive allows a user to specifiy a timeout on proxy requests. This is usefull when you have a slow/buggy appserver which hangs, and you would rather just return a timeout and fail gracefully instead of waiting however long it takes the server to return

ProxyVia Directive

Description:	Information provided in the Via HTTP response header for proxied requests
Syntax:	`ProxyVia on|off|full|block`
Default:	`ProxyVia off`
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

This directive controls the use of the `Via:` HTTP header by the proxy. Its intended use is to control the flow of of proxy requests along a chain of proxy servers. See RFC2068 (HTTP/1.1) for an explanation of `Via:` header lines.

- If set to *off*, which is the default, no special processing is performed. If a request or reply contains a `Via:` header, it is passed through unchanged.
- If set to *on*, each request and reply will get a `Via:` header line added for the current host.
- If set to *full*, each generated `Via:` header line will additionally have the Apache server version shown as a `Via:` comment field.
- If set to *block*, every proxy request will have all its `Via:` header lines removed. No new `Via:` header will be generated.

URI References

[1] http://www.squid.org
[2] http://www.squid-cache.org/
[3] http://httpd.apache.org/docs-2.1/dns-caveats.html