56(\*\*

Modules | Directives | FAQ | Glossary | Sitemap Apache HTTP Server Version 2.5 <- Apache > HTTP Server > Documentation > Version 2.5 > Modules Apache Module mod\_proxy Available Languages: en | fr | ja Description:Multi-protocol proxy/gateway server Status:Extension Module Identifier:proxy\_module Source File:mod\_proxy.c Summary Warning 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. mod\_proxy and related modules implement a proxy/gateway for Apache HTTP Server, supporting a number of popular protocols as well as several different load balancing algorithms. Third-party modules can add support for additional protocols and load balancing algorithms. A set of modules must be loaded into the server to provide the necessary features. These modules can be included statically at build time or dynamically via the LoadModule directive). The set must include: mod\_proxy, which provides basic proxy capabilities mod\_proxy\_balancer and one or more balancer modules if load balancing is required. (See mod\_proxy\_balancer for more information.) one or more proxy scheme, or protocol, modules: ProtocolModule AJP13 (Apache JServe Protocol version 1.3)mod\_proxy\_ajp CONNECT (for SSL)mod\_proxy\_connect FastCGImod\_proxy\_fcgi ftpmod\_proxy\_ftp HTTP/0.9, HTTP/1.0, and HTTP/1.1mod\_proxy\_http HTTP/2.0mod\_proxy\_http2 SCGImod\_proxy\_scgi UWSGImod\_proxy\_uwsgi WS and WSS (Web-sockets)mod\_proxy\_wstunnel In addition, extended features are provided by other modules. Caching is provided by mod\_cache and related modules. The ability to contact remote servers using the SSL/TLS protocol is provided by the SSLProxy\* directives of mod\_ssl. These additional modules will need to be loaded and configured to take advantage of these features. Topics Forward Proxies and Reverse Proxies/Gateways Basic Examples Access via Handler Workers Controlling Access to Your Proxy Slow Startup Intranet Proxy Protocol Adjustments Request Bodies Reverse Proxy Request Headers Directives BalancerGrowth BalancerInherit BalancerMember BalancerPersist NoProxy Proxy100Continue ProxyAddHeaders ProxyBadHeader ProxyBlock ProxyDomain ProxyErrorOverride ProxyIOBufferSize ProxyMaxForwards ProxyPass ProxyPassInherit ProxyPassInterpolateEnv ProxyPassMatch ProxyPassReverse ProxyPassReverseCookieDomain ProxyPassReverseCookiePath ProxyPreserveHost ProxyReceiveBufferSize ProxyRemote ProxyRemoteMatch ProxyRequests ProxySet ProxySourceAddress ProxyStatus ProxyTimeout ProxyVia Bugfix checklist httpd changelog Known issues Report a bug See also mod\_cache mod\_proxy\_ajp mod\_proxy\_balancer mod\_proxy\_connect mod\_proxy\_fcgi mod\_proxy\_ftp mod\_proxy\_hcheck mod\_proxy\_http mod\_proxy\_scgi mod\_proxy\_wstunnel mod\_ssl Comments top Forward Proxies and Reverse Proxies/Gateways Apache HTTP Server can be configured in both a forward and reverse proxy (also known as gateway) mode. An ordinary forward proxy is an intermediate server that sits between the client and the origin server. In order to get content from the origin server, the client sends a request to the proxy naming the origin server as the target. The proxy then requests the content from the origin server and returns it to the client. The client must be specially configured to use the forward proxy to access other sites. A typical usage of a forward proxy is to provide Internet access to internal clients that are otherwise restricted by a firewall. The forward proxy can also use caching (as provided by mod\_cache) to reduce network usage. The forward proxy is activated using the ProxyRequests directive. Because forward proxies allow clients to access arbitrary sites through your server and to hide their true origin, it is essential that you secure your server so that only authorized clients can access the proxy before activating a forward proxy. A reverse proxy (or gateway), by contrast, appears to the client just like an ordinary web server. No special configuration on the client is necessary. The client make ordinary requests for content in the namespace of the reverse proxy. The reverse proxy then decides where to send those requests and returns the content as if it were itself the origin. A typical usage of a reverse proxy is to provide Internet users access to a server that is behind a firewall. Reverse proxies can also be used to balance load among several back-end servers or to provide caching for a slower back-end server. In addition, reverse proxies can be used simply to bring several servers into the same URL space. A reverse proxy is activated using the ProxyPass directive or the [P] flag to the RewriteRule directive. It is not necessary to turn ProxyRequests on in order to configure a reverse proxy. top Basic Examples The examples below are only a very basic idea to help you get started. Please read the documentation on the individual directives. In addition, if you wish to have caching enabled, consult the documentation from mod\_cache. Reverse Proxy ProxyPass "/foo" "http://foo.example.com/bar" ProxyPassReverse "/foo" "http://foo.example.com/bar" Forward Proxy ProxyRequests On ProxyVia On Require host internal.example.com top Access via Handler You can also force a request to be handled as a reverse-proxy request, by creating a suitable Handler pass-through. The example configuration below will pass all requests for PHP scripts to the specified FastCGI server using reverse proxy: Reverse Proxy PHP scripts SetHandler "proxy:unix:/path/to/app.sock|fcgi://localhost/" This feature is available in Apache HTTP Server 2.4.10 and later. top Workers The proxy manages the configuration of origin servers and their communication parameters in objects called workers. There are two built-in workers: the default forward proxy worker and the default reverse proxy worker. Additional workers can be configured explicitly. The two default workers have a fixed configuration and will be used if no other worker matches the request. They do not use HTTP Keep-Alive or connection reuse. The TCP connections to the origin server will instead be opened and closed for each request. Explicitly configured workers are identified by their URL. They are usually created and configured using ProxyPass or ProxyPassMatch when used for a reverse proxy: ProxyPass "/example" "http://backend.example.com" connectiontimeout=5 timeout=30 This will create a worker associated with the origin server URL http://backend.example.com that will use the given timeout values. All timeouts use the time-interval directive syntax. When used in a forward proxy, workers are usually defined via the ProxySet directive: ProxySet http://backend.example.com connectiontimeout=5 timeout=30 or alternatively using Proxy and ProxySet: ProxySet connectiontimeout=5 timeout=30 Using explicitly configured workers in the forward mode is not very common, because forward proxies usually communicate with many different origin servers. Creating explicit workers for some of the origin servers can still be useful if they are used very often. Explicitly configured workers have no concept of forward or reverse proxying by themselves. They encapsulate a common concept of communication with origin servers. A worker created by ProxyPass for use in a reverse proxy will also be used for forward proxy requests whenever the URL to the origin server matches the worker URL, and vice versa. The URL identifying a direct worker is the URL of its origin server including any path components given: ProxyPass "/examples" "http://backend.example.com/examples" ProxyPass "/docs" "http://backend.example.com/docs" This example defines two different workers, each using a separate connection pool and configuration. Worker Sharing Worker sharing happens if the worker URLs overlap, which occurs when the URL of some worker is a leading substring of the URL of another worker defined later in the configuration file. In the following example ProxyPass "/apps" "http://backend.example.com/" timeout=60 ProxyPass "/examples" "http://backend.example.com/examples" timeout=10 the second worker isn't actually created. Instead the first worker is used. The benefit is, that there is only one connection pool, so connections are more often reused. Note that all configuration attributes given explicitly for the later worker will be ignored. This will be logged as a warning. In the above example, the resulting timeout value for the URL /examples will be 60 instead of 10! If you want to avoid worker sharing, sort your worker definitions by URL length, starting with the longest worker URLs. If you want to maximize worker sharing, use the reverse sort order. See also the related warning about ordering ProxyPass directives. Host part in the URL The host part needs to start with a letter [a-z]. For example: ProxyPass "/apps" "http://127" is not valid and will cause an error while processing a request that maps the path. Explicitly configured workers come in two flavors: direct workers and (load) balancer workers. They support many important configuration attributes which are described below in the ProxyPass directive. The same attributes can also be set using ProxySet. The set of options available for a direct worker depends on the protocol which is specified in the origin server URL. Available protocols include ajp, fcgi, ftp, http and scgi. Balancer workers are virtual workers that use direct workers known as their members to actually handle the requests. Each balancer can have multiple members. When it handles a request, it chooses a member based on the configured load balancing algorithm. A balancer worker is created if its worker URL uses balancer as the protocol scheme. The balancer URL uniquely identifies the balancer worker. Members are added to a balancer using BalancerMember. DNS resolution for origin domains DNS resolution happens when the socket to the origin domain is created for the first time. When connection reuse is enabled, each backend domain is resolved only once per child process, and cached for all further connections until the child is recycled. This information should to be considered while planning DNS maintenance tasks involving backend domains. Please also check ProxyPass parameters for more details about connection reuse. top Controlling Access to Your Proxy You can control who can access your proxy via the control block as in the following example: Require ip 192.168.0 For more information on access control directives, see mod\_authz\_host. Strictly limiting access is essential if you are using a forward proxy (using the ProxyRequests directive). Otherwise, your server can be used by any client to access arbitrary hosts while hiding his or her true identity. This is dangerous both for your network and for the Internet at large. When using a reverse proxy (using the ProxyPass directive with ProxyRequests Off), access control is less critical because clients can only contact the hosts that you have specifically configured. See Also the Proxy-Chain-Auth environment variable. top Slow Startup 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. top Intranet Proxy An Apache httpd proxy server situated in an intranet needs to forward external requests through the company's firewall (for this, configure the ProxyRemote directive to forward the respective scheme to the firewall proxy). 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.example.com/. 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 httpd 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. top Protocol Adjustments For circumstances where mod\_proxy is sending requests to an origin server that doesn't properly implement keepalives or HTTP/1.1, there are two environment variables that can force the request to use 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. ProxyPass "http://buggyappserver:7001/foo/" SetEnv force-proxy-request-1.0 1 SetEnv proxy-nokeepalive 1 In 2.4.26 and later, the "no-proxy" environment variable can be set to disable mod\_proxy processing the current request. This variable should be set with SetEnvIf, as SetEnv is not evaluated early enough. top Request Bodies Some request methods such as POST include a request body. The HTTP protocol requires that requests which include a body either use chunked transfer encoding or send a Content-Length request header. When passing these requests on to the origin server, mod\_proxy\_http will always attempt to send the Content-Length. But if the body is large and the original request used chunked encoding, then chunked encoding may also be used in the upstream request. You can control this selection using environment variables. Setting proxy-sendcl ensures maximum compatibility with upstream servers by always sending the Content-Length, while setting proxy-sendchunked minimizes resource usage by using chunked encoding. Under some circumstances, the server must spool request bodies to disk to satisfy the requested handling of request bodies. For example, this spooling will occur if the original body was sent with chunked encoding (and is large), but the administrator has asked for backend requests to be sent with Content-Length or as HTTP/1.0. This spooling can also occur if the request body already has a Content-Length header, but the server is configured to filter incoming request bodies. LimitRequestBody only applies to request bodies that the server will spool to disk top Reverse Proxy Request Headers When acting in a reverse-proxy mode (using the ProxyPass directive, for example), mod\_proxy\_http adds several request headers in order to pass information to the origin server. These headers are: X-Forwarded-For The IP address of the client. X-Forwarded-Host The original host requested by the client in the Host HTTP request header. X-Forwarded-Server The hostname of the proxy server. Be careful when using these headers on the origin server, since they will contain more than one (comma-separated) value if the original request already contained one of these headers. For example, you can use %{X-Forwarded-For}i in the log format string of the origin server to log the original clients IP address, but you may get more than one address if the request passes through several proxies. See also the ProxyPreserveHost and ProxyVia directives, which control other request headers. Note: If you need to specify custom request headers to be added to the forwarded request, use the RequestHeader directive. top BalancerGrowth Directive Description:Number of additional Balancers that can be added Post-configuration Syntax:BalancerGrowth # Default:BalancerGrowth 5 Context:server config, virtual host Status:Extension Module:mod\_proxy Compatibility:BalancerGrowth is only available in Apache HTTP Server 2.3.13 and later. This directive allows for growth potential in the number of Balancers available for a virtualhost in addition to the number pre-configured. It only takes effect if there is at least one pre-configured Balancer. top BalancerInherit Directive Description:Inherit proxy Balancers/Workers defined from the main server Syntax:BalancerInherit On|Off Default:BalancerInherit On Context:server config, virtual host Status:Extension Module:mod\_proxy Compatibility:BalancerInherit is only available in Apache HTTP Server 2.4.5 and later. This directive will cause the current server/vhost to "inherit" Balancers and Workers defined in the main server. This can cause issues and inconsistent behavior if using the Balancer Manager for dynamic changes and so should be disabled if using that feature. The setting in the global server defines the default for all vhosts. Disabling ProxyPassInherit also disables BalancerInherit. top BalancerMember Directive Description:Add a member to a load balancing group Syntax:BalancerMember [balancerurl] url [key=value [key=value ...]] Context:directory Status:Extension Module:mod\_proxy This directive adds a member to a load balancing group. It can be used within a container directive and can take any of the key value pair parameters available to ProxyPass directives. One additional parameter is available only to BalancerMember directives: loadfactor. This is the member load factor - a decimal between 1.0 (default) and 100.0, which defines the weighted load to be applied to the member in question. The balancerurl is only needed when not within a container directive. It corresponds to the url of a balancer defined in ProxyPass directive. The path component of the balancer URL in any container directive is ignored. Trailing slashes should typically be removed from the URL of a BalancerMember. top BalancerPersist Directive Description:Attempt to persist changes made by the Balancer Manager across restarts. Syntax:BalancerPersist On|Off Default:BalancerPersist Off Context:server config, virtual host Status:Extension Module:mod\_proxy Compatibility:BalancerPersist is only available in Apache HTTP Server 2.4.4 and later. This directive will cause the shared memory storage associated with the balancers and balancer members to be persisted across restarts. This allows these local changes to not be lost during the normal restart/graceful state transitions. top 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 httpd 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.example.com:81 NoProxy .example.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 .example.org. To distinguish Domains from Hostnames (both syntactically and semantically; a DNS domain can have a DNS A record, too!), Domains are always written with a leading period. Note Domain name comparisons are done without regard to the case, and Domains are always assumed to be anchored in the root of the DNS tree; therefore, the two domains .ExAmple.com and .example.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 IPAddrs via the DNS domain name service. It represents a logical host (in contrast to Domains, see above) and must be resolvable to at least one IPAddr (or often to a list of hosts with different IPAddrs). Examples prep.ai.example.edu www.example.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 httpd can take a remarkable deal of time when the connection to the name server uses a slow PPP link. Hostname comparisons are done without regard to the case, and Hostnames are always assumed to be anchored in the root of the DNS tree; therefore, the two hosts WWW.ExAmple.com and www.example.com. (note the trailing period) are considered equal. See also DNS Issues top Directive Description:Container for directives applied to proxied resources Syntax: ... Context:server config, virtual host Status:Extension Module:mod\_proxy Directives placed in 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: Require host yournetwork.example.com 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: SetOutputFilter INCLUDES The next example will allow web clients from the specified IP addresses to issue CONNECT requests to access the https://www.example.com/ SSL server if mod\_proxy\_connect is enabled. Require ip 192.168.0.0/16 Differences from the Location configuration section A backend URL matches the configuration section if it begins with the the wildcard-url string, even if the last path segment in the directive only matches a prefix of the backend URL. For example, matches all of http://example.com/foo, http://example.com/foo/bar, and http://example.com/foobar. The matching of the final URL differs from the behavior of the section, which for purposes of this note treats the final path component as if it ended in a slash. For more control over the matching, see . See also top Proxy100Continue Directive Description:Forward 100-continue expectation to the origin server Syntax:Proxy100Continue Off|On Default:Proxy100Continue On Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Compatibility:Available in version 2.4.40 and later This directive determines whether the proxy should forward 100-continue Expect:ation to the origin server and thus let it decide when/if the HTTP request body should be read, or when Off the proxy should generate 100 Continue intermediate response by itself before forwarding the request body. Effectiveness This option is of use only for HTTP proxying, as handled by mod\_proxy\_http. top ProxyAddHeaders Directive Description:Add proxy information in X-Forwarded-\* headers Syntax:ProxyAddHeaders Off|On Default:ProxyAddHeaders On Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Compatibility:Available in version 2.3.10 and later This directive determines whether or not proxy related information should be passed to the backend server through X-Forwarded-For, X-Forwarded-Host and X-Forwarded-Server HTTP headers. Effectiveness This option is of use only for HTTP proxying, as handled by mod\_proxy\_http. top ProxyBadHeader Directive Description:Determines how to handle bad header lines in a response Syntax:ProxyBadHeader IsError|Ignore|StartBody Default:ProxyBadHeader IsError Context:server config, virtual host Status:Extension Module:mod\_proxy The ProxyBadHeader directive determines the behavior of mod\_proxy if it receives syntactically invalid response header lines (i.e. containing no colon) from the origin server. The following arguments are possible: IsError Abort the request and end up with a 502 (Bad Gateway) response. This is the default behavior. Ignore Treat bad header lines as if they weren't sent. StartBody When receiving the first bad header line, finish reading the headers and treat the remainder as body. This helps to work around buggy backend servers which forget to insert an empty line between the headers and the body. top ProxyBlock Directive Description:Disallow proxy requests to certain hosts Syntax:ProxyBlock \*|hostname|partial-hostname [hostname|partial-hostname]... Context:server config, virtual host Status:Extension Module:mod\_proxy The ProxyBlock directive can be used to block FTP or HTTP access to certain hosts via the proxy, based on a full or partial hostname match, or, if applicable, an IP address comparison. Each of the arguments to the ProxyBlock directive can be either \* or a alphanumeric string. At startup, the module will attempt to resolve every alphanumeric string from a DNS name to a set of IP addresses, but any DNS errors are ignored. If an asterisk "\*" argument is specified, mod\_proxy will deny access to all FTP or HTTP sites. Otherwise, for any request for an HTTP or FTP resource via the proxy, mod\_proxy will check the hostname of the request URI against each specified string. If a partial string match is found, access is denied. If no matches against hostnames are found, and a remote (forward) proxy is configured using ProxyRemote or ProxyRemoteMatch, access is allowed. If no remote (forward) proxy is configured, the IP address of the hostname from the URI is compared against all resolved IP addresses determined at startup. Access is denied if any match is found. Note that the DNS lookups may slow down the startup time of the server. Example ProxyBlock news.example.com auctions.example.com friends.example.com Note that example would also be sufficient to match any of these sites. Hosts would also be matched if referenced by IP address. Note also that ProxyBlock \* blocks connections to all sites. top 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 httpd 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.example.com:81" NoProxy ".example.com" "192.168.112.0/21" ProxyDomain ".example.com" top ProxyErrorOverride Directive Description:Override error pages for proxied content Syntax:ProxyErrorOverride Off|On [code ...] Default:ProxyErrorOverride Off Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Compatibility:The list of status codes was added in 2.5.1 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.) This directive does not affect the processing of informational (1xx), normal success (2xx), or redirect (3xx) responses. By default ProxyErrorOverride affects all responses with codes between 400 (including) and 600 (excluding). Example for default behavior ProxyErrorOverride On To change the default behavior, you can specify the status codes to consider, separated by spaces. If you do so, all other status codes will be ignored. You can only specify status codes, that are considered error codes: between 400 (including) and 600 (excluding). Example for custom status codes ProxyErrorOverride On 403 405 500 501 502 503 504 top ProxyIOBufferSize Directive Description:Determine size of internal data throughput buffer Syntax:ProxyIOBufferSize bytes Default:ProxyIOBufferSize 8192 Context:server config, virtual host Status:Extension Module:mod\_proxy The ProxyIOBufferSize directive adjusts the size of the internal buffer which is used as a scratchpad for the data between input and output. The size must be at least 512. In almost every case, there's no reason to change that value. If used with AJP, this directive sets the maximum AJP packet size in bytes. Values larger than 65536 are set to 65536. If you change it from the default, you must also change the packetSize attribute of your AJP connector on the Tomcat side! The attribute packetSize is only available in Tomcat 5.5.20+ and 6.0.2+ Normally it is not necessary to change the maximum packet size. Problems with the default value have been reported when sending certificates or certificate chains. top Directive Description:Container for directives applied to regular-expression-matched proxied resources Syntax: ... Context:server config, virtual host Status:Extension Module:mod\_proxy The directive is identical to the directive, except that it matches URLs using regular expressions. From 2.4.8 onwards, named groups and backreferences are captured and written to the environment with the corresponding name prefixed with "MATCH\_" and in upper case. This allows elements of URLs to be referenced from within expressions and modules like mod\_rewrite. In order to prevent confusion, numbered (unnamed) backreferences are ignored. Use named groups instead. [^/]+)> require ldap-group cn=%{env:MATCH\_SITENAME},ou=combined,o=Example See also top ProxyMaxForwards Directive Description:Maximum number of proxies that a request can be forwarded through Syntax:ProxyMaxForwards number Default:ProxyMaxForwards -1 Context:server config, virtual host Status:Extension Module:mod\_proxy The ProxyMaxForwards directive specifies the maximum number of proxies through which a request may pass if there's no Max-Forwards header supplied with the request. This may be set to prevent infinite proxy loops or a DoS attack. Example ProxyMaxForwards 15 Note that setting ProxyMaxForwards is a violation of the HTTP/1.1 protocol (RFC2616), which forbids a Proxy setting Max-Forwards if the Client didn't set it. Earlier Apache httpd versions would always set it. A negative ProxyMaxForwards value, including the default -1, gives you protocol-compliant behavior but may leave you open to loops. top ProxyPass Directive Description:Maps remote servers into the local server URL-space Syntax:ProxyPass [path] !|url [key=value [key=value ...]] [nocanon] [interpolate] [noquery] Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Compatibility:Unix Domain Socket (UDS) support added in 2.4.7 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. The local server is often called a reverse proxy or gateway. The path is the name of a local virtual path; url is a partial URL for the remote server and cannot include a query string. It is strongly suggested to review the concept of a Worker before proceeding any further with this section. This directive is not supported within , and containers. The ProxyRequests directive should usually be set off when using ProxyPass. In 2.4.7 and later, support for using a Unix Domain Socket is available by using a target which prepends unix:/path/lis.sock|. For example, to proxy HTTP and target the UDS at /home/www.socket, you would use unix:/home/www.socket|http://localhost/whatever/. Since the socket is local, the hostname used (in this case localhost) is moot, but it is passed as the Host: header value of the request. Note: The path associated with the unix: URL is DefaultRuntimeDir aware. Note: RewriteRule requires the [P,NE] option to prevent the '|' character from being escaped. When used inside a section, the first argument is omitted and the local directory is obtained from the . The same will occur inside a section; however, ProxyPass does not interpret the regexp as such, so it is necessary to use ProxyPassMatch in this situation instead. Suppose the local server has address http://example.com/; then ProxyPass "http://backend.example.com/" will cause a local request for http://example.com/mirror/foo/bar to be internally converted into a proxy request to http://backend.example.com/bar. If you require a more flexible reverse-proxy configuration, see the RewriteRule directive with the [P] flag. The following alternative syntax is possible; however, it can carry a performance penalty when present in very large numbers. The advantage of the below syntax is that it allows for dynamic control via the Balancer Manager interface: ProxyPass "/mirror/foo/" "http://backend.example.com/" If the first argument ends with a trailing /, the second argument should also end with a trailing /, and vice versa. Otherwise, the resulting requests to the backend may miss some needed slashes and do not deliver the expected results. The ! directive is useful in situations where you don't want to reverse-proxy a subdirectory, e.g. ProxyPass "http://backend.example.com/" ProxyPass "!" ProxyPass "/mirror/foo/i" "!" ProxyPass "/mirror/foo" "http://backend.example.com" will proxy all requests to /mirror/foo to backend.example.com except requests made to /mirror/foo/i. Mixing ProxyPass settings in different contexts does not work: ProxyPass "/mirror/foo/i" "!" ProxyPass "http://backend.example.com/" In this case, a request to /mirror/foo/i will get proxied, because the ProxyPass directive in the Location block will be evaluated first. The fact that ProxyPass supports both server and directory contexts does not mean that their scope and position in the configuration file will guarantee any ordering or override. Ordering ProxyPass Directives The configured ProxyPass and ProxyPassMatch rules are checked in the order of configuration. The first rule that matches wins. So usually you should sort conflicting ProxyPass rules starting with the longest URLs first. Otherwise, later rules for longer URLS will be hidden by any earlier rule which uses a leading substring of the URL. Note that there is some relation with worker sharing. Ordering ProxyPass Directives in Locations Only one ProxyPass directive can be placed in a Location block, and the most specific location will take precedence. Exclusions and the no-proxy environment variable Exclusions must come before the general ProxyPass directives. In 2.4.26 and later, the "no-proxy" environment variable is an alternative to exclusions, and is the only way to configure an exclusion of a ProxyPass directive in Location context. This variable should be set with SetEnvIf, as SetEnv is not evaluated early enough. ProxyPass key=value Parameters In Apache HTTP Server 2.1 and later, mod\_proxy supports pooled connections to a backend server. Connections created on demand can be retained in a pool for future use. Limits on the pool size and other settings can be coded on the ProxyPass directive using key=value parameters, described in the tables below. Maximum connections to the backend By default, mod\_proxy will allow and retain the maximum number of connections that could be used simultaneously by that web server child process. Use the max parameter to reduce the number from the default. The pool of connections is maintained per web server child process, and max and other settings are not coordinated among all child processes, except when only one child process is allowed by configuration or MPM design. Use the ttl parameter to set an optional time to live; connections which have been unused for at least ttl seconds will be closed. ttl can be used to avoid using a connection which is subject to closing because of the backend server's keep-alive timeout. Example ProxyPass "/example" "http://backend.example.com" max=20 ttl=120 retry=300 Worker|BalancerMember parameters ParameterDefaultDescription min0Minimum number of connection pool entries, unrelated to the actual number of connections. This only needs to be modified from the default for special circumstances where heap memory associated with the backend connections should be preallocated or retained. max1...nMaximum number of connections that will be allowed to the backend server. The default for this limit is the number of threads per process in the active MPM. In the Prefork MPM, this is always 1; while with other MPMs, it is controlled by the ThreadsPerChild directive. smaxmaxRetained connection pool entries above this limit are freed during certain operations if they have been unused for longer than the time to live, controlled by the ttl parameter. If the connection pool entry has an associated connection, it will be closed. This only needs to be modified from the default for special circumstances where connection pool entries and any associated connections which have exceeded the time to live need to be freed or closed more aggressively. acquire-If set, this will be the maximum time to wait for a free connection in the connection pool, in milliseconds. If there are no free connections in the pool, the Apache httpd will return SERVER\_BUSY status to the client. connectiontimeouttimeoutConnect timeout in seconds. The number of seconds Apache httpd waits for the creation of a connection to the backend to complete. By adding a postfix of ms, the timeout can be also set in milliseconds. Uses the time-interval directive syntax disablereuseOffThis parameter should be used when you want to force mod\_proxy to immediately close a connection to the backend after being used, and thus, disable its persistent connection and pool for that backend. This helps in various situations where a firewall between Apache httpd and the backend server (regardless of protocol) tends to silently drop connections or when backends themselves may be under round- robin DNS. When connection reuse is enabled each backend domain is resolved (with a DNS query) only once per child process and cached for all further connections until the child is recycled. To disable connection reuse, set this property value to On. enablereuseOnThis is the inverse of 'disablereuse' above, provided as a convenience for scheme handlers that require opt-in for connection reuse (such as mod\_proxy\_fcgi). flushpacketsoffDetermines whether the proxy module will auto-flush the output brigade after each "chunk" of data. 'off' means that it will flush only when needed; 'on' means after each chunk is sent; and 'auto' means poll/wait for a period of time and flush if no input has been received for 'flushwait' milliseconds. Currently, this is in effect only for mod\_proxy\_ajp and mod\_proxy\_fcgi. flushwait10The time to wait for additional input, in milliseconds, before flushing the output brigade if 'flushpackets' is 'auto'. Uses time-interval directive syntax. iobuffersize8192Adjusts the size of the internal scratchpad IO buffer. This allows you to override the ProxyIOBufferSize for a specific worker. This must be at least 512 or set to 0 for the system default of 8192. responsefieldsize8192Adjust the size of the proxy response field buffer. The buffer size should be at least the size of the largest expected header size from a proxied response. Setting the value to 0 will use the system default of 8192 bytes. Available in Apache HTTP Server 2.4.34 and later. keepaliveOff This parameter should be used when you have a firewall between your Apache httpd and the backend server, which tends to drop inactive connections. This flag will tell the Operating System to send KEEP\_ALIVE messages on inactive connections and thus prevent the firewall from dropping the connection. To enable keepalive, set this property value to On. The frequency of initial and subsequent TCP keepalive probes depends on global OS settings, and may be as high as 2 hours. To be useful, the frequency configured in the OS must be smaller than the threshold used by the firewall. Uses the time-interval directive syntax. lbset0Sets the load balancer cluster set that the worker is a member of. The load balancer will try all members of a lower numbered lbset before trying higher numbered ones. ping0Ping property tells the webserver to "test" the connection to the backend before forwarding the request. For negative values, the test is a simple socket check; for positive values, it's a more functional check, dependent upon the protocol. For AJP, it causes mod\_proxy\_ajp to send a CPING request on the ajp13 connection (implemented on Tomcat 3.3.2+, 4.1.28+ and 5.0.13+). For HTTP, it causes mod\_proxy\_http to send a 100-Continue to the backend (only valid for HTTP/1.1 - for non HTTP/1.1 backends, this property has no effect). In both cases, the parameter is the delay in seconds to wait for the reply. This feature has been added to avoid problems with hung and busy backends. This will increase the network traffic during the normal operation which could be an issue, but it will lower the traffic in case some of the cluster nodes are down or busy. By adding a postfix of ms, the delay can be also set in milliseconds. Uses the time-interval directive syntax. receivebuffersize0Adjusts the size of the explicit (TCP/IP) network buffer size for proxied connections. This allows you to override the ProxyReceiveBufferSize for a specific worker. This must be at least 512 or set to 0 for thes system default. redirect-Redirection Route of the worker. This value is usually set dynamically to enable safe removal of the node from the cluster. If set, all requests without session id will be redirected to the BalancerMember that has route parameter equal to this value. retry60Connection pool worker retry timeout in seconds. If the connection pool worker to the backend server is in the error state, Apache httpd will not forward any requests to that server until the timeout expires. This enables to shut down the backend server for maintenance and bring it back online later. A value of 0 means always retry workers in an error state with no timeout. Uses the time-interval directive syntax. route-Route of the worker when used inside load balancer. The route is a value appended to session id. status-Single letter value defining the initial status of this worker. DWorker is disabled and will not accept any requests; will be automatically retried. SWorker is administratively stopped; will not accept requests and will not be automatically retried. IWorker is in ignore-errors mode and will always be considered available. RWorker is a hot spare. For each worker in a given lbset that is unusable (draining, stopped, in error, etc.), a usable hot spare with the same lbset will be used in its place. Hot spares can help ensure that a specific number of workers are always available for use by a balancer. HWorker is in hot-standby mode and will only be used if no other viable workers or spares are available in the balancer set. EWorker is in an error state. NWorker is in drain mode and will only accept existing sticky sessions destined for itself and ignore all other requests. Status can be set (which is the default) by prepending with '+' or cleared by prepending with '-'. Thus, a setting of 'S-E' sets this worker to Stopped and clears the in-error flag. timeoutProxyTimeoutConnection timeout in seconds. The number of seconds Apache httpd waits for data sent by / to the backend. Uses the time-interval directive syntax. ttl-Time to live for inactive connections and associated connection pool entries, in seconds. Once reaching this limit, a connection will not be used again; it will be closed at some later time. Uses the time-interval directive syntax. flusherflush Name of the provider used by mod\_proxy\_fdpass. See the documentation of this module for more details. secret- Value of secret used by mod\_proxy\_ajp. See the documentation of this module for more details. upgradeWebSocket Protocol accepted in the Upgrade header by mod\_proxy\_wstunnel. See the documentation of this module for more details. mapping- Type of mapping between the path and the url. This determines the normalization and/or (non-)decoding that mod\_proxy will apply to the requested uri-path before matching the path. If a mapping matches, it's committed to the uri-path such that all the directory contexts that use a path (like ) will be matched using the same mapping. mapping=encoded prevents the %-decoding of the uri-path so that one can use for instance configurations like: ProxyPass "/special%3Fsegment" "https://example.com/special%3Fsegment" mapping=encoded Require ip 172.17.2.0/24 mapping=servlet refers to the normalization defined by the Servlet specification, which is for instance applied by Apache Tomcat for servlet containers (notably the path parameters are ignored for the mapping). An uri-path like /some;foo/path is then mapped as /some/path hence matches any of the below regardless of the requested path parameters: ProxyPass "/some/path" "https://servlet.example.com/some/path" mapping=servlet Require valid-user Note It is recommended to use the same mapping on the Apache httpd side than the one used on the backend side. For instance when configuring authorizations in blocks for paths that are mapped by mod\_proxy to some servlet containers (like applications running on Apache Tomcat), one should use the mapping=servlet setting to prevent path parameters and alike from interfering with the authorizations that are to be enforced in by the Apache httpd. If the Proxy directive scheme starts with the balancer:// (eg: balancer://cluster, any path information is ignored), then a virtual worker that does not really communicate with the backend server will be created. Instead, it is responsible for the management of several "real" workers. In that case, the special set of parameters can be added to this virtual worker. See mod\_proxy\_balancer for more information about how the balancer works. Balancer parameters ParameterDefaultDescription lbmethodbyrequestsBalancer load-balance method. Select the load-balancing scheduler method to use. Either byrequests, to perform weighted request counting; bytraffic, to perform weighted traffic byte count balancing; or bybusyness, to perform pending request balancing. The default is byrequests. maxattemptsOne less than the number of workers, or 1 with a single worker.Maximum number of failover attempts before giving up. nofailoverOffIf set to On, the session will break if the worker is in error state or disabled. Set this value to On if backend servers do not support session replication. stickysession-Balancer sticky session name. The value is usually set to something like JSESSIONID or PHPSESSIONID, and it depends on the backend application server that support sessions. If the backend application server uses different names for cookies and url encoded id (like servlet containers), use | to separate them. The first part is for the cookie; the second is for the path. Available in Apache HTTP Server 2.4.4 and later. stickysessionsep"."Sets the separation symbol in the session cookie. Some backend application servers do not use the '.' as the symbol. For example, the Oracle Weblogic server uses '!'. The correct symbol can be set using this option. The setting of 'Off' signifies that no symbol is used. scolonpathdelimOffIf set to On, the semi-colon character ';' will be used as an additional sticky session path delimiter/separator. This is mainly used to emulate mod\_jk's behavior when dealing with paths such as JSESSIONID=6736bcf34;foo=aabfa timeout0Balancer timeout in seconds. If set, this will be the maximum time to wait for a free worker. The default is to not wait. Uses the time-interval directive syntax. failonstatus-A single or comma-separated list of HTTP status codes. If set, this will force the worker into error state when the backend returns any status code in the list. Worker recovery behaves the same as other worker errors. failontimeoutOffIf set, an IO read timeout after a request is sent to the backend will force the worker into error state. Worker recovery behaves the same as other worker errors. Available in Apache HTTP Server 2.4.5 and later. nonceThe protective nonce used in the balancer-manager application page. The default is to use an automatically determined UUID-based nonce, to provide for further protection for the page. If set, then the nonce is set to that value. A setting of None disables all nonce checking. Note In addition to the nonce, the balancer-manager page should be protected via an ACL. growth0Number of additional BalancerMembers to allow to be added to this balancer in addition to those defined at configuration. forcerecoveryOnForce the immediate recovery of all workers without considering the retry parameter of the workers if all workers of a balancer are in error state. There might be cases where an already overloaded backend can get into deeper trouble if the recovery of all workers is enforced without considering the retry parameter of each worker. In this case, set to Off. Available in Apache HTTP Server 2.4.2 and later. A sample balancer setup: ProxyPass "/special-area" "http://special.example.com" smax=5 max=10 ProxyPass "/" "balancer://mycluster/" stickysession=JSESSIONID|jsessionid nofailover=On BalancerMember ajp://1.2.3.4:8009 BalancerMember ajp://1.2.3.5:8009 loadfactor=20 # Less powerful server, don't send as many requests there, BalancerMember ajp://1.2.3.6:8009 loadfactor=5 Configuring hot spares can help ensure that a certain number of workers are always available for use per load balancer set: ProxyPass "/" "balancer://sparecluster/" BalancerMember ajp://1.2.3.4:8009 BalancerMember ajp://1.2.3.5:8009 # The servers below are hot spares. For each server above that is unusable # (draining, stopped, unreachable, in error state, etc.), one of these spares # will be used in its place. Two servers will always be available for a request # unless one or more of the spares is also unusable. BalancerMember ajp://1.2.3.6:8009 status=+R BalancerMember ajp://1.2.3.7:8009 status=+R Setting up a hot-standby that will only be used if no other members (or spares) are available in the load balancer set: ProxyPass "/" "balancer://hotcluster/" BalancerMember ajp://1.2.3.4:8009 loadfactor=1 BalancerMember ajp://1.2.3.5:8009 loadfactor=2.25 # The server below is on hot standby BalancerMember ajp://1.2.3.6:8009 status=+H ProxySet lbmethod=bytraffic Additional ProxyPass Keywords Normally, mod\_proxy will canonicalise ProxyPassed URLs. But this may be incompatible with some backends, particularly those that make use of PATH\_INFO. The optional nocanon keyword suppresses this and passes the URL path "raw" to the backend. Note that this keyword may affect the security of your backend, as it removes the normal limited protection against URL-based attacks provided by the proxy. Normally, mod\_proxy will include the query string when generating the SCRIPT\_FILENAME environment variable. The optional noquery keyword (available in httpd 2.4.1 and later) prevents this. The optional interpolate keyword, in combination with ProxyPassInterpolateEnv, causes the ProxyPass to interpolate environment variables, using the syntax ${VARNAME}. Note that many of the standard CGI-derived environment variables will not exist when this interpolation happens, so you may still have to resort to mod\_rewrite for complex rules. Also note that interpolation is supported within the scheme/hostname/port portion of a URL only for variables that are available when the directive is parsed (like Define). Dynamic determination of those fields can be accomplished with mod\_rewrite. The following example describes how to use mod\_rewrite to dynamically set the scheme to http or https: RewriteEngine On RewriteCond %{HTTPS} =off RewriteRule . - [E=protocol:http] RewriteCond %{HTTPS} =on RewriteRule . - [E=protocol:https] RewriteRule ^/mirror/foo/(.\*) %{ENV:protocol}://backend.example.com/$1 [P] ProxyPassReverse "/mirror/foo/" "http://backend.example.com/" ProxyPassReverse "/mirror/foo/" "https://backend.example.com/" top ProxyPassInherit Directive Description:Inherit ProxyPass directives defined from the main server Syntax:ProxyPassInherit On|Off Default:ProxyPassInherit On Context:server config, virtual host Status:Extension Module:mod\_proxy Compatibility:ProxyPassInherit is only available in Apache HTTP Server 2.4.5 and later. This directive will cause the current server/vhost to "inherit" ProxyPass directives defined in the main server. This can cause issues and inconsistent behavior if using the Balancer Manager for dynamic changes and so should be disabled if using that feature. The setting in the global server defines the default for all vhosts. Disabling ProxyPassInherit also disables BalancerInherit. top ProxyPassInterpolateEnv Directive Description:Enable Environment Variable interpolation in Reverse Proxy configurations Syntax:ProxyPassInterpolateEnv On|Off Default:ProxyPassInterpolateEnv Off Context:server config, virtual host, directory Status:Extension Module:mod\_proxy This directive, together with the interpolate argument to ProxyPass, ProxyPassReverse, ProxyPassReverseCookieDomain, and ProxyPassReverseCookiePath, enables reverse proxies to be dynamically configured using environment variables which may be set by another module such as mod\_rewrite. It affects the ProxyPass, ProxyPassReverse, ProxyPassReverseCookieDomain, and ProxyPassReverseCookiePath directives and causes them to substitute the value of an environment variable varname for the string ${varname} in configuration directives if the interpolate option is set. The scheme/hostname/port portion of ProxyPass may contain variables, but only the ones available when the directive is parsed (for example, using Define). For all the other use cases, please consider using mod\_rewrite instead. Performance warning Keep this turned off unless you need it! Adding variables to ProxyPass for example may lead to the use of the default mod\_proxy's workers configured (that don't allow any fine tuning like connections reuse, etc..). top ProxyPassMatch Directive Description:Maps remote servers into the local server URL-space using regular expressions Syntax:ProxyPassMatch [regex] !|url [key=value [key=value ...]] Context:server config, virtual host, directory Status:Extension Module:mod\_proxy This directive is equivalent to ProxyPass but makes use of regular expressions instead of simple prefix matching. The supplied regular expression is matched against the url, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a new url. Note: This directive cannot be used within a context. Suppose the local server has address http://example.com/; then ProxyPassMatch "^/(.\*\.gif)$" "http://backend.example.com/$1" will cause a local request for http://example.com/foo/bar.gif to be internally converted into a proxy request to http://backend.example.com/foo/bar.gif. Note The URL argument must be parsable as a URL before regexp substitutions (as well as after). This limits the matches you can use. For instance, if we had used ProxyPassMatch "^(/.\*\.gif)$" "http://backend.example.com:8000$1" in our previous example, it would fail with a syntax error at server startup. This is a bug (PR 46665 in the ASF bugzilla), and the workaround is to reformulate the match: ProxyPassMatch "^/(.\*\.gif)$" "http://backend.example.com:8000/$1" The ! directive is useful in situations where you don't want to reverse-proxy a subdirectory. When used inside a section, the first argument is omitted and the regexp is obtained from the . If you require a more flexible reverse-proxy configuration, see the RewriteRule directive with the [P] flag. Default Substitution When the URL parameter doesn't use any backreferences into the regular expression, the original URL will be appended to the URL parameter. Security Warning Take care when constructing the target URL of the rule, considering the security impact from allowing the client influence over the set of URLs to which your server will act as a proxy. Ensure that the scheme and hostname part of the URL is either fixed or does not allow the client undue influence. top ProxyPassReverse Directive Description:Adjusts the URL in HTTP response headers sent from a reverse proxied server Syntax:ProxyPassReverse [path] url [interpolate] Context:server config, virtual host, directory Status:Extension Module:mod\_proxy This directive lets Apache httpd adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache httpd is used as a reverse proxy (or gateway) to avoid bypassing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy. Only the HTTP response headers specifically mentioned above will be rewritten. Apache httpd will not rewrite other response headers, nor will it by default rewrite URL references inside HTML pages. This means that if the proxied content contains absolute URL references, they will bypass the proxy. To rewrite HTML content to match the proxy, you must load and enable mod\_proxy\_html. path is the name of a local virtual path. It is combined with the requests scheme, host, and port to form the replacement prefix of the affected response header. This parameter is usually the same as the first parameter of the ProxyPass directive. url is a partial URL for the remote server. When the remote server sends the specifically mentioned headers beginning with this partial URL, the prefix from the remote server is replaced with a prefix that points to the reverse proxy. This parameter is usually the same as the second parameter of the ProxyPass directive, but it may differ if the remote server uses a different scheme, host, port, or path in the affected headers. In configurations with ProxyPreserveHost, the hostname in the partial URL may already be the hostname of the reverse proxy, which will prevent the matching and replacement done by this directive. For example, suppose the local server has address http://example.com/; then ProxyPass "/mirror/foo/" "http://backend.example.com/" ProxyPassReverse "/mirror/foo/" "http://backend.example.com/" ProxyPassReverseCookieDomain backend.example.com public.example.com ProxyPassReverseCookiePath "/" "/mirror/foo/" will not only cause a local request for the http://example.com/mirror/foo/bar to be internally converted into a proxy request to http://backend.example.com/bar (the functionality which ProxyPass provides here). It also takes care of redirects which the server backend.example.com sends when redirecting http://backend.example.com/bar to http://backend.example.com/quux . Apache httpd adjusts this to http://example.com/mirror/foo/quux before forwarding the HTTP redirect response to the client. Note that the hostname used for constructing the URL is chosen in respect to the setting of the UseCanonicalName directive. Note that this ProxyPassReverse directive can also be used in conjunction with the proxy feature (RewriteRule ... [P]) from mod\_rewrite because it doesn't depend on a corresponding ProxyPass directive. The optional interpolate keyword, used together with ProxyPassInterpolateEnv, enables interpolation of environment variables specified using the format ${VARNAME}. Note that interpolation is not supported within the scheme portion of a URL. When used inside a section, the first argument is omitted and the local directory is obtained from the . The same occurs inside a section, but will probably not work as intended, as ProxyPassReverse will interpret the regexp literally as a path; if needed in this situation, specify the ProxyPassReverse outside the section or in a separate section. This directive is not supported in or sections. top ProxyPassReverseCookieDomain Directive Description:Adjusts the Domain string in Set-Cookie headers from a reverse- proxied server Syntax:ProxyPassReverseCookieDomain internal-domain public-domain [interpolate] Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Usage is basically similar to ProxyPassReverse, but instead of rewriting headers that are a URL, this rewrites the domain string in Set-Cookie headers. top ProxyPassReverseCookiePath Directive Description:Adjusts the Path string in Set-Cookie headers from a reverse- proxied server Syntax:ProxyPassReverseCookiePath internal-path public-path [interpolate] Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Useful in conjunction with ProxyPassReverse in situations where backend URL paths are mapped to public paths on the reverse proxy. This directive rewrites the path string in Set-Cookie headers. If the beginning of the cookie path matches internal-path, the cookie path will be replaced with public-path. In the example given with ProxyPassReverse, the directive: ProxyPassReverseCookiePath "/" "/mirror/foo/" will rewrite a cookie with backend path / (or /example or, in fact, anything) to /mirror/foo/. top ProxyPreserveHost Directive Description:Use incoming Host HTTP request header for proxy request Syntax:ProxyPreserveHost On|Off Default:ProxyPreserveHost Off Context:server config, virtual host, directory Status:Extension Module:mod\_proxy Compatibility:Usable in directory context in 2.3.3 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. It is mostly useful in special configurations like proxied mass name-based virtual hosting, where the original Host header needs to be evaluated by the backend server. top ProxyReceiveBufferSize Directive Description:Network buffer size for proxied HTTP and FTP connections Syntax:ProxyReceiveBufferSize bytes Default:ProxyReceiveBufferSize 0 Context:server config, virtual host Status:Extension Module:mod\_proxy The ProxyReceiveBufferSize directive specifies an explicit (TCP/IP) network buffer size for proxied 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 top ProxyRemote Directive Description:Remote proxy used to handle certain requests Syntax:ProxyRemote match remote-server [username:password] Context:server config, virtual host Status:Extension Module:mod\_proxy Compatibility:The optional third argument is usable only in httpd 2.5.1 and later. 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 = scheme://hostname[:port] scheme is effectively the protocol that should be used to communicate with the remote server; only http and https are supported by this module. When using https, the requests are forwarded through the remote proxy using the HTTP CONNECT method. Example ProxyRemote http://goodguys.example.com/ http://mirrorguys.example.com:8000 ProxyRemote \* http://cleverproxy.localdomain ProxyRemote ftp http://ftpproxy.mydomain: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. An optional third argument username:password may be given, which defines the Basic authentication credentials to pass to the configured remote proxy. The credentials will always be sent without first waiting for the remote proxy to send a Basic authentication challenge. The Proxy-Chain-Auth environment variable has no effect if this argument is used. top ProxyRemoteMatch Directive Description:Remote proxy used to handle requests matched by regular expressions Syntax:ProxyRemoteMatch regex remote-server Context:server config, virtual host Status:Extension Module:mod\_proxy The ProxyRemoteMatch is identical to the ProxyRemote directive, except that the first argument is a regular expression match against the requested URL. top 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 httpd from functioning as a forward proxy server. (Setting ProxyRequests to Off does not disable use of the ProxyPass directive.) In a typical reverse proxy or gateway configuration, this option should be set to Off. In order to get the functionality of proxying HTTP or FTP sites, you need also mod\_proxy\_http or mod\_proxy\_ftp (or both) present in the server. In order to get the functionality of (forward) proxying HTTPS sites, you need mod\_proxy\_connect enabled in the server. Warning 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. See also Forward and Reverse Proxies/Gateways top ProxySet Directive Description:Set various Proxy balancer or member parameters Syntax:ProxySet url key=value [key=value ...] Context:server config, virtual host, directory Status:Extension Module:mod\_proxy This directive is used as an alternate method of setting any of the parameters available to Proxy balancers and workers normally done via the ProxyPass directive. If used within a container directive, the url argument is not required. As a side effect the respective balancer or worker gets created. This can be useful when doing reverse proxying via a RewriteRule instead of a ProxyPass directive. BalancerMember http://www2.example.com:8080 loadfactor=1 BalancerMember http://www3.example.com:8080 loadfactor=2 ProxySet lbmethod=bytraffic ProxySet keepalive=On ProxySet balancer://foo lbmethod=bytraffic timeout=15 ProxySet ajp://backend:7001 timeout=15 Warning Keep in mind that the same parameter key can have a different meaning depending whether it is applied to a balancer or a worker, as shown by the two examples above regarding timeout. top ProxySourceAddress Directive Description:Set local IP address for outgoing proxy connections Syntax:ProxySourceAddress address Context:server config, virtual host Status:Extension Module:mod\_proxy Compatibility:Available in version 2.3.9 and later This directive allows to set a specific local address to bind to when connecting to a backend server. top ProxyStatus Directive Description:Show Proxy LoadBalancer status in mod\_status Syntax:ProxyStatus Off|On|Full Default:ProxyStatus Off Context:server config, virtual host Status:Extension Module:mod\_proxy This directive determines whether or not proxy loadbalancer status data is displayed via the mod\_status server-status page. Note Full is synonymous with On top ProxyTimeout Directive Description:Network timeout for proxied requests Syntax:ProxyTimeout time-interval[s] Default:Value of Timeout Context:server config, virtual host Status:Extension Module:mod\_proxy This directive allows a user to specify a timeout on proxy requests. This is useful 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. top 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 proxy requests along a chain of proxy servers. See RFC 2616 (HTTP/1.1), section 14.45 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 httpd 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. Available Languages: en | fr | ja top Comments Notice: This is not a Q&A section. Comments placed here should be pointed towards suggestions on improving the documentation or server, and may be removed by our moderators if they are either implemented or considered invalid/off-topic. Questions on how to manage the Apache HTTP Server should be directed at either our SRS channels Ip index 30A, #httpd, on Libera.chat, or sent to our mailing lists. Copyright 2021 The Apache Software Foundation. Licensed under the Apache License, Version 2.0.Modules | Directives | FAQ | Glossary | Sitemap

E+