7. The Main Configuration File

By default, the main configuration file is named `config`, with the exception of Windows, where it is named `config.txt`. Configuration lines consist of an initial keyword followed by a list of values, all separated by whitespace (any number of spaces or tabs). For example:

```
confdir /etc/privoxy
```

Assigns the value `/etc/privoxy` to the option `confdir` and thus indicates that the configuration directory is named "/etc/privoxy/".

All options in the config file except for `confdir` and `logdir` are optional. Watch out in the below description for what happens if you leave them unset.

The main config file controls all aspects of Privoxy's operation that are not location dependent (i.e. they apply universally, no matter where you may be surfing). Like the filter and action files, the config file is a plain text file and can be modified with a text editor like emacs, vim or notepad.exe.

7.1. Local Set-up Documentation

If you intend to operate Privoxy for more users than just yourself, it might be a good idea to let them know how to reach you, what you block and why you do that, your policies, etc.

7.1.1. user-manual

Specifies:

Location of the Privoxy User Manual.

Type of value:

A fully qualified URI

Default value:

Unset

Effect if unset:

https://www.privoxy.org/*version*/user-manual/ will be used, where *version* is the Privoxy version.

Notes:

The User Manual URI is the single best source of information on Privoxy, and is used for help links from some of the internal CGI pages. The manual itself is normally packaged with the binary distributions, so you probably want to set this to a locally installed copy.

Examples:

The best all purpose solution is simply to put the full local `PATH` to where the *User Manual* is located:

```
user-manual  /usr/share/doc/privoxy/user-manual
```

The User Manual is then available to anyone with access to Privoxy, by following the built-in URL: `http://config.privoxy.org/user-manual/` (or the shortcut: `http://p.p/user-manual/`).

If the documentation is not on the local system, it can be accessed from a remote server, as:

```
user-manual  http://example.com/privoxy/user-manual/
```

Warning
If set, this option should be *the first option in the config file*, because it is used while the config file is being read on start-up.

7.1.2. trust-info-url

Specifies:

A URL to be displayed in the error page that users will see if access to an untrusted page is denied.

Type of value:

URL

Default value:

Unset

Effect if unset:

No links are displayed on the "untrusted" error page.

Notes:

The value of this option only matters if the experimental trust mechanism has been activated. (See *trustfile* below.)

If you use the trust mechanism, it is a good idea to write up some on-line documentation about your trust policy and to specify the URL(s) here. Use multiple times for multiple URLs.

The URL(s) should be added to the trustfile as well, so users don't end up locked out from the information on why they were locked out in the first place!

7.1.3. admin-address

Specifies:

An email address to reach the Privoxy administrator.

Type of value:

Email address

Default value:

Unset

Effect if unset:

No email address is displayed on error pages and the CGI user interface.

Notes:

If both `admin-address` and `proxy-info-url` are unset, the whole "Local Privoxy Support" box on all generated pages will not be shown.

7.1.4. proxy-info-url

Specifies:

A URL to documentation about the local Privoxy setup, configuration or policies.

Type of value:

URL

Default value:

Unset

Effect if unset:

No link to local documentation is displayed on error pages and the CGI user interface.

Notes:

If both `admin-address` and `proxy-info-url` are unset, the whole "Local Privoxy Support" box on all generated pages will not be shown.

This URL shouldn't be blocked ;-)

7.2. Configuration and Log File Locations

Privoxy can (and normally does) use a number of other files for additional configuration, help and logging. This section of the configuration file tells Privoxy where to find those other files.

The user running Privoxy, must have read permission for all configuration files, and write permission to any files that would be modified, such as log files and actions files.

7.2.1. confdir

Specifies:

The directory where the other configuration files are located.

Type of value:

Path name

Default value:

/etc/privoxy (Unix) *or* Privoxy installation dir (Windows)

Effect if unset:

Mandatory

Notes:

No trailing "`/`", please.

7.2.2. templdir

Specifies:

An alternative directory where the templates are loaded from.

Type of value:

Path name

Default value:

unset

Effect if unset:

The templates are assumed to be located in confdir/template.

Notes:

Privoxy's original templates are usually overwritten with each update. Use this option to relocate customized templates that should be kept. As template variables might change between updates, you shouldn't expect templates to work with Privoxy releases other than the one they were part of, though.

7.2.3. temporary-directory

Specifies:

A directory where Privoxy can create temporary files.

Type of value:

Path name

Default value:

unset

Effect if unset:

No temporary files are created, external filters don't work.

Notes:

To execute `external filters`, Privoxy has to create temporary files. This directive specifies the directory the temporary files should be written to.

It should be a directory only Privoxy (and trusted users) can access.

7.2.4. logdir

Specifies:

The directory where all logging takes place (i.e. where the `logfile` is located).

Type of value:

Path name

Default value:

/var/log/privoxy (Unix) *or* Privoxy installation dir (Windows)

Effect if unset:

Mandatory

Notes:

No trailing "/", please.

7.2.5. actionsfile

Specifies:

The actions file(s) to use

Type of value:

Complete file name, relative to `confdir`

Default values:

```
match-all.action # Actions that are applied to all sites and maybe overruled later on.

default.action   # Main actions file

user.action      # User customizations
```

Effect if unset:

No actions are taken at all. More or less neutral proxying.

Notes:

Multiple `actionsfile` lines are permitted, and are in fact recommended!

The default values are `default.action`, which is the "main" actions file maintained by the developers, and `user.action`, where you can make your personal additions.

Actions files contain all the per site and per URL configuration for ad blocking, cookie management, privacy considerations, etc.

7.2.6. filterfile

Specifies:

The filter file(s) to use

Type of value:

File name, relative to `confdir`

Default value:

default.filter (Unix) *or* default.filter.txt (Windows)

Effect if unset:

No textual content filtering takes place, i.e. all `+filter{`*`name`*`}` actions in the actions files are turned neutral.

Notes:

Multiple `filterfile` lines are permitted.

The filter files contain content modification rules that use regular expressions. These rules permit powerful changes on the content of Web pages, and optionally the headers as well, e.g., you could try to disable your favorite JavaScript annoyances, re-write the actual displayed text, or just have some fun playing buzzword bingo with web pages.

The `+filter{`*`name`*`}` actions rely on the relevant filter (*`name`*) to be defined in a filter file!

A pre-defined filter file called `default.filter` that contains a number of useful filters for common problems is included in the distribution. See the section on the `filter` action for a list.

It is recommended to place any locally adapted filters into a separate file, such as `user.filter`.

7.2.7. logfile

Specifies:

The log file to use

Type of value:

File name, relative to `logdir`

Default value:

Unset (commented out). When activated: logfile (Unix) *or* privoxy.log (Windows).

Effect if unset:

No logfile is written.

Notes:

The logfile is where all logging and error messages are written. The level of detail and number of messages are set with the `debug` option (see below). The logfile can be useful for tracking down a problem with Privoxy (e.g., it's not blocking an ad you think it should block) and it can help you to monitor what your browser is doing.

Depending on the debug options below, the logfile may be a privacy risk if third parties can get access to it. As most users will never look at it, Privoxy only logs fatal errors by default.

For most troubleshooting purposes, you will have to change that, please refer to the debugging section for details.

Any log files must be writable by whatever user Privoxy is being run as (on Unix, default user id is "privoxy").

To prevent the logfile from growing indefinitely, it is recommended to periodically rotate or shorten it. Many operating systems support log rotation out of the box, some require additional software to do it. For details, please refer to the documentation for your operating system.

7.2.8. trustfile

Specifies:

The name of the trust file to use

Type of value:

File name, relative to `confdir`

Default value:

Unset (commented out). When activated: trust (Unix) *or* trust.txt (Windows)

Effect if unset:

The entire trust mechanism is disabled.

Notes:

The trust mechanism is an experimental feature for building white-lists and should be used with care. It is *NOT* recommended for the casual user.

If you specify a trust file, Privoxy will only allow access to sites that are specified in the trustfile. Sites can be listed in one of two ways:

Prepending a `~` character limits access to this site only (and any sub-paths within this site), e.g. `~www.example.com` allows access to `~www.example.com/features/news.html`, etc.

Or, you can designate sites as *trusted referrers*, by prepending the name with a `+` character. The effect is that access to untrusted sites will be granted -- but only if a link from this trusted referrer was used to get there. The link target will then be added to the "trustfile" so that future, direct accesses will be granted. Sites added via this mechanism do not become trusted referrers themselves (i.e. they are added with a `~` designation). There is a limit of 512 such entries, after which new entries will not be made.

If you use the `+` operator in the trust file, it may grow considerably over time.

It is recommended that Privoxy be compiled with the `--disable-force`, `--disable-toggle` and `--disable-editor` options, if this feature is to be used.

Possible applications include limiting Internet access for children.

7.3. Debugging

These options are mainly useful when tracing a problem. Note that you might also want to invoke Privoxy with the `--no-daemon` command line option when debugging.

7.3.1. debug

Specifies:

Key values that determine what information gets logged.

Type of value:

Integer values

Default value:

0 (i.e.: only fatal errors (that cause Privoxy to exit) are logged)

Effect if unset:

Default value is used (see above).

Notes:

The available debug levels are:

```
debug     1 # Log the destination for each request Privoxy let through. See also debug 1024.
debug     2 # show each connection status
debug     4 # show I/O status
debug     8 # show header parsing
debug    16 # log all data written to the network
debug    32 # debug force feature
debug    64 # debug regular expression filters
debug   128 # debug redirects
debug   256 # debug GIF de-animation
debug   512 # Common Log Format
debug  1024 # Log the destination for requests Privoxy didn't let through, and the reason why.
debug  2048 # CGI user interface
debug  4096 # Startup banner and warnings.
debug  8192 # Non-fatal errors
debug 32768 # log all data read from the network
debug 65536 # Log the applying actions
```

To select multiple debug levels, you can either add them or use multiple `debug` lines.

A debug level of 1 is informative because it will show you each request as it happens. *1, 1024, 4096 and 8192 are recommended* so that you will notice when things go wrong. The other levels are probably only of interest if you are hunting down a specific problem. They can produce a hell of an output (especially 16).

If you are used to the more verbose settings, simply enable the debug lines below again.

If you want to use pure CLF (Common Log Format), you should set "debug 512" *ONLY* and not enable anything else.

Privoxy has a hard-coded limit for the length of log messages. If it's reached, messages are logged truncated and marked with "... [too long, truncated]".

Please don't file any support requests without trying to reproduce the problem with increased debug level first. Once you read the log messages, you may even be able to solve the problem on your own.

7.3.2. single-threaded

Specifies:

Whether to run only one server thread.

Type of value:

1 or 0

Default value:

0

Effect if unset:

Multi-threaded (or, where unavailable: forked) operation, i.e. the ability to serve multiple requests simultaneously.

Notes:

This option is only there for debugging purposes. *It will drastically reduce performance.*

7.3.3. hostname

Specifies:

The hostname shown on the CGI pages.

Type of value:

Text

Default value:

Unset

Effect if unset:

The hostname provided by the operating system is used.

Notes:

On some misconfigured systems resolving the hostname fails or takes too much time and slows Privoxy down. Setting a fixed hostname works around the problem.

In other circumstances it might be desirable to show a hostname other than the one returned by the operating system. For example if the system has several different hostnames and you don't want to use the first one.

Note that Privoxy does not validate the specified hostname value.

7.4. Access Control and Security

This section of the config file controls the security-relevant aspects of Privoxy's configuration.

7.4.1. listen-address

Specifies:

The address and TCP port on which Privoxy will listen for client requests.

Type of value:

[*IP-Address*]:*Port*

[*Hostname*]:*Port*

Default value:

127.0.0.1:8118

Effect if unset:

Bind to 127.0.0.1 (IPv4 localhost), port 8118. This is suitable and recommended for home users who run Privoxy on the same machine as their browser.

Notes:

You will need to configure your browser(s) to this proxy address and port.

If you already have another service running on port 8118, or if you want to serve requests from other machines (e.g. on your local network) as well, you will need to override the default.

You can use this statement multiple times to make Privoxy listen on more ports or more IP addresses. Suitable if your operating system does not support sharing IPv6 and IPv4 protocols on the same socket.

If a hostname is used instead of an IP address, Privoxy will try to resolve it to an IP address

and if there are multiple, use the first one returned.

If the address for the hostname isn't already known on the system (for example because it's in /etc/hostname), this may result in DNS traffic.

If the specified address isn't available on the system, or if the hostname can't be resolved, Privoxy will fail to start.

IPv6 addresses containing colons have to be quoted by brackets. They can only be used if Privoxy has been compiled with IPv6 support. If you aren't sure if your version supports it, have a look at `http://config.privoxy.org/show-status`.

Some operating systems will prefer IPv6 to IPv4 addresses even if the system has no IPv6 connectivity which is usually not expected by the user. Some even rely on DNS to resolve localhost which mean the "localhost" address used may not actually be local.

It is therefore recommended to explicitly configure the intended IP address instead of relying on the operating system, unless there's a strong reason not to.

If you leave out the address, Privoxy will bind to all IPv4 interfaces (addresses) on your machine and may become reachable from the Internet and/or the local network. Be aware that some GNU/Linux distributions modify that behaviour without updating the documentation. Check for non-standard patches if your Privoxy version behaves differently.

If you configure Privoxy to be reachable from the network, consider using access control lists (ACL's, see below), and/or a firewall.

If you open Privoxy to untrusted users, you will also want to make sure that the following actions are disabled: `enable-edit-actions` and `enable-remote-toggle`

Example:

Suppose you are running Privoxy on a machine which has the address 192.168.0.1 on your local private network (192.168.0.0) and has another outside connection with a different address. You want it to serve requests from inside only:

```
listen-address  192.168.0.1:8118
```

Suppose you are running Privoxy on an IPv6-capable machine and you want it to listen on the IPv6 address of the loopback device:

```
listen-address [::1]:8118
```

7.4.2. toggle

Specifies:

Initial state of "toggle" status

Type of value:

1 or 0

Default value:

1

Effect if unset:

Act as if toggled on

Notes:

If set to 0, Privoxy will start in "toggled off" mode, i.e. mostly behave like a normal, content-neutral proxy with both ad blocking and content filtering disabled. See `enable-remote-toggle` below.

7.4.3. enable-remote-toggle

Specifies:

Whether or not the web-based toggle feature may be used

Type of value:

0 or 1

Default value:

0

Effect if unset:

The web-based toggle feature is disabled.

Notes:

When toggled off, Privoxy mostly acts like a normal, content-neutral proxy, i.e. doesn't block ads or filter content.

Access to the toggle feature can *not* be controlled separately by "ACLs" or HTTP authentication, so that everybody who can access Privoxy (see "ACLs" and `listen-address` above) can toggle it for all users. So this option is *not recommended* for multi-user environments with untrusted users.

Note that malicious client side code (e.g Java) is also capable of using this option.

As a lot of Privoxy users don't read documentation, this feature is disabled by default.

Note that you must have compiled Privoxy with support for this feature, otherwise this option has no effect.

7.4.4. enable-remote-http-toggle

Specifies:

Whether or not Privoxy recognizes special HTTP headers to change its behaviour.

Type of value:

0 or 1

Default value:

0

Effect if unset:

Privoxy ignores special HTTP headers.

Notes:

When toggled on, the client can change Privoxy's behaviour by setting special HTTP headers. Currently the only supported special header is "X-Filter: No", to disable filtering for the ongoing request, even if it is enabled in one of the action files.

This feature is disabled by default. If you are using Privoxy in a environment with trusted clients, you may enable this feature at your discretion. Note that malicious client side code (e.g Java) is also capable of using this feature.

This option will be removed in future releases as it has been obsoleted by the more general header taggers.

7.4.5. enable-edit-actions

Specifies:

Whether or not the web-based actions file editor may be used

Type of value:

0 or 1

Default value:

0

Effect if unset:

The web-based actions file editor is disabled.

Notes:

Access to the editor can *not* be controlled separately by "ACLs" or HTTP authentication, so that everybody who can access Privoxy (see "ACLs" and `listen-address` above) can modify its configuration for all users.

This option is *not recommended* for environments with untrusted users and as a lot of Privoxy users don't read documentation, this feature is disabled by default.

Note that malicious client side code (e.g Java) is also capable of using the actions editor and you shouldn't enable this options unless you understand the consequences and are sure your browser is configured correctly.

Note that you must have compiled Privoxy with support for this feature, otherwise this option has no effect.

7.4.6. enforce-blocks

Specifies:

Whether the user is allowed to ignore blocks and can "go there anyway".

Type of value:

`0 or 1`

Default value:

0

Effect if unset:

Blocks are not enforced.

Notes:

Privoxy is mainly used to block and filter requests as a service to the user, for example to block ads and other junk that clogs the pipes. Privoxy's configuration isn't perfect and sometimes innocent pages are blocked. In this situation it makes sense to allow the user to enforce the request and have Privoxy ignore the block.

In the default configuration Privoxy's "Blocked" page contains a "go there anyway" link to adds a special string (the force prefix) to the request URL. If that link is used, Privoxy will detect the force prefix, remove it again and let the request pass.

Of course Privoxy can also be used to enforce a network policy. In that case the user obviously should not be able to bypass any blocks, and that's what the "enforce-blocks" option is for. If it's enabled, Privoxy hides the "go there anyway" link. If the user adds the force prefix by hand, it will not be accepted and the circumvention attempt is logged.

Examples:

enforce-blocks 1

7.4.7. ACLs: permit-access and deny-access

Specifies:

Who can access what.

Type of value:

src_addr[:*port*][/*src_masklen*] [*dst_addr*[:*port*][/*dst_masklen*]]

Where *src_addr* and *dst_addr* are IPv4 addresses in dotted decimal notation or valid DNS names, *port* is a port number, and *src_masklen* and *dst_masklen* are subnet masks in CIDR notation, i.e. integer values from 2 to 30 representing the length (in bits) of the network address. The masks and the whole destination part are optional.

If your system implements RFC 3493, then *src_addr* and *dst_addr* can be IPv6 addresses delimeted by brackets, *port* can be a number or a service name, and *src_masklen* and *dst_masklen* can be a number from 0 to 128.

Default value:

Unset

If no *port* is specified, any port will match. If no *src_masklen* or *src_masklen* is given, the complete IP address has to match (i.e. 32 bits for IPv4 and 128 bits for IPv6).

Effect if unset:

Don't restrict access further than implied by `listen-address`

Notes:

Access controls are included at the request of ISPs and systems administrators, and *are not usually needed by individual users*. For a typical home user, it will normally suffice to ensure that Privoxy only listens on the localhost (127.0.0.1) or internal (home) network address by means of the *listen-address* option.

Please see the warnings in the FAQ that Privoxy is not intended to be a substitute for a firewall or to encourage anyone to defer addressing basic security weaknesses.

Multiple ACL lines are OK. If any ACLs are specified, Privoxy only talks to IP addresses that match at least one `permit-access` line and don't match any subsequent `deny-access` line. In other words, the last match wins, with the default being `deny-access`.

If Privoxy is using a forwarder (see `forward` below) for a particular destination URL, the *`dst_addr`* that is examined is the address of the forwarder and *NOT* the address of the ultimate target. This is necessary because it may be impossible for the local Privoxy to determine the IP address of the ultimate target (that's often what gateways are used for).

You should prefer using IP addresses over DNS names, because the address lookups take time. All DNS names must resolve! You can *not* use domain patterns like "*.org" or partial domain names. If a DNS name resolves to multiple IP addresses, only the first one is used.

Some systems allow IPv4 clients to connect to IPv6 server sockets. Then the client's IPv4 address will be translated by the system into IPv6 address space with special prefix ::ffff:0:0/96 (so called IPv4 mapped IPv6 address). Privoxy can handle it and maps such ACL addresses automatically.

Denying access to particular sites by ACL may have undesired side effects if the site in question is hosted on a machine which also hosts other sites (most sites are).

Examples:

Explicitly define the default behavior if no ACL and `listen-address` are set: "localhost" is OK. The absence of a *`dst_addr`* implies that *all* destination addresses are OK:

```
permit-access  localhost
```

Allow any host on the same class C subnet as www.privoxy.org access to nothing but www.example.com (or other domains hosted on the same system):

```
permit-access  www.privoxy.org/24 www.example.com/32
```

Allow access from any host on the 26-bit subnet 192.168.45.64 to anywhere, with the exception that 192.168.45.73 may not access the IP address behind www.dirty-stuff.example.com:

```
permit-access  192.168.45.64/26
deny-access    192.168.45.73    www.dirty-stuff.example.com
```

Allow access from the IPv4 network 192.0.2.0/24 even if listening on an IPv6 wild card address (not supported on all platforms):

```
permit-access  192.0.2.0/24
```

This is equivalent to the following line even if listening on an IPv4 address (not supported on all platforms):

```
permit-access  [::ffff:192.0.2.0]/120
```

7.4.8. buffer-limit

Specifies:

Maximum size of the buffer for content filtering.

Type of value:

Size in Kbytes

Default value:

4096

Effect if unset:

Use a 4MB (4096 KB) limit.

Notes:

For content filtering, i.e. the `+filter` and `+deanimate-gif` actions, it is necessary that Privoxy buffers the entire document body. This can be potentially dangerous, since a server could just keep sending data indefinitely and wait for your RAM to exhaust -- with nasty consequences. Hence this option.

When a document buffer size reaches the `buffer-limit`, it is flushed to the client unfiltered and no further attempt to filter the rest of the document is made. Remember that there may be multiple threads running, which might require up to `buffer-limit` Kbytes *each*, unless you have enabled "single-threaded" above.

7.4.9. enable-proxy-authentication-forwarding

Specifies:

Whether or not proxy authentication through Privoxy should work.

Type of value:

0 or 1

Default value:

0

Effect if unset:

Proxy authentication headers are removed.

Notes:

Privoxy itself does not support proxy authentication, but can allow clients to authenticate against Privoxy's parent proxy.

By default Privoxy (3.0.21 and later) don't do that and remove Proxy-Authorization headers in requests and Proxy-Authenticate headers in responses to make it harder for malicious sites to trick inexperienced users into providing login information.

If this option is enabled the headers are forwarded.

Enabling this option is *not recommended* if there is no parent proxy that requires authentication or if the local network between Privoxy and the parent proxy isn't trustworthy. If proxy authentication is only required for some requests, it is recommended to use a client header filter to remove the authentication headers for requests where they aren't needed.

7.5. Forwarding

This feature allows routing of HTTP requests through a chain of multiple proxies.

Forwarding can be used to chain Privoxy with a caching proxy to speed up browsing. Using a parent proxy may also be necessary if the machine that Privoxy runs on has no direct Internet access.

Note that parent proxies can severely decrease your privacy level. For example a parent proxy could add your IP address to the request headers and if it's a caching proxy it may add the "Etag" header to revalidation requests again, even though you configured Privoxy to remove it. It may also ignore Privoxy's header time randomization and use the original values which could be used by the server as cookie replacement to track your steps between visits.

Also specified here are SOCKS proxies. Privoxy supports the SOCKS 4 and SOCKS 4A protocols.

7.5.1. forward

Specifies:

To which parent HTTP proxy specific requests should be routed.

Type of value:

target_pattern http_parent[:*port*]

where *target_pattern* is a URL pattern that specifies to which requests (i.e. URLs) this forward rule shall apply. Use `/` to denote "all URLs". *http_parent*[:*port*] is the DNS name or IP address of the parent HTTP proxy through which the requests should be forwarded, optionally followed by its listening port (default: 8000). Use a single dot (`.`) to denote "no forwarding".

Default value:

Unset

Effect if unset:

Don't use parent HTTP proxies.

Notes:

If *http_parent* is ".", then requests are not forwarded to another HTTP proxy but are made directly to the web servers.

http_parent can be a numerical IPv6 address (if RFC 3493 is implemented). To prevent clashes with the port delimiter, the whole IP address has to be put into brackets. On the other hand a *target_pattern* containing an IPv6 address has to be put into angle brackets (normal brackets are reserved for regular expressions already).

Multiple lines are OK, they are checked in sequence, and the last match wins.

Examples:

Everything goes to an example parent proxy, except SSL on port 443 (which it doesn't handle):

```
forward   /      parent-proxy.example.org:8080
forward   :443   .
```

Everything goes to our example ISP's caching proxy, except for requests to that ISP's sites:

```
forward   /                  caching-proxy.isp.example.net:8000
forward   .isp.example.net   .
```

Parent proxy specified by an IPv6 address:

```
forward   /                  [2001:DB8::1]:8000
```

Suppose your parent proxy doesn't support IPv6:

```
forward  /                       parent-proxy.example.org:8000
forward  ipv6-server.example.org  .
forward  <[2-3][0-9a-f][0-9a-f][0-9a-f]:*>   .
```

7.5.2. forward-socks4, forward-socks4a, forward-socks5 and forward-socks5t

Specifies:

Through which SOCKS proxy (and optionally to which parent HTTP proxy) specific requests should be routed.

Type of value:

target_pattern socks_proxy[:*port*] *http_parent*[:*port*]

where *target_pattern* is a URL pattern that specifies to which requests (i.e. URLs) this forward rule shall apply. Use `/` to denote "all URLs". *http_parent* and *socks_proxy* are IP addresses in dotted decimal notation or valid DNS names (*http_parent* may be "." to denote "no HTTP forwarding"), and the optional *port* parameters are TCP ports, i.e. integer values from 1 to 65535

Default value:

Unset

Effect if unset:

Don't use SOCKS proxies.

Notes:

Multiple lines are OK, they are checked in sequence, and the last match wins.

The difference between `forward-socks4` and `forward-socks4a` is that in the SOCKS 4A protocol, the DNS resolution of the target hostname happens on the SOCKS server, while in SOCKS 4 it happens locally.

With `forward-socks5` the DNS resolution will happen on the remote server as well.

`forward-socks5t` works like vanilla `forward-socks5` but lets Privoxy additionally use Tor-specific SOCKS extensions. Currently the only supported SOCKS extension is optimistic data which can reduce the latency for the first request made on a newly created connection.

socks_proxy and *http_parent* can be a numerical IPv6 address (if RFC 3493 is implemented). To prevent clashes with the port delimiter, the whole IP address has to be put into brackets. On the other hand a *target_pattern* containing an IPv6 address has to be put into angle brackets (normal brackets are reserved for regular expressions already).

If *http_parent* is ".", then requests are not forwarded to another HTTP proxy but are made (HTTP-wise) directly to the web servers, albeit through a SOCKS proxy.

Examples:

From the company example.com, direct connections are made to all "internal" domains, but

everything outbound goes through their ISP's proxy by way of example.com's corporate SOCKS 4A gateway to the Internet.

```
forward-socks4a   /              socks-gw.example.com:1080  www-cache.isp.example.net:8080
forward           .example.com   .
```

A rule that uses a SOCKS 4 gateway for all destinations but no HTTP parent looks like this:

```
forward-socks4   /              socks-gw.example.com:1080  .
```

To chain Privoxy and Tor, both running on the same system, you would use something like:

```
forward-socks5t   /               127.0.0.1:9050 .
```

Note that if you got Tor through one of the bundles, you may have to change the port from 9050 to 9150 (or even another one). For details, please check the documentation on the Tor website.

The public Tor network can't be used to reach your local network, if you need to access local servers you therefore might want to make some exceptions:

```
forward         192.168.*.*/     .
forward            10.*.*.*/     .
forward           127.*.*.*/     .
```

Unencrypted connections to systems in these address ranges will be as (un)secure as the local network is, but the alternative is that you can't reach the local network through Privoxy at all. Of course this may actually be desired and there is no reason to make these exceptions if you aren't sure you need them.

If you also want to be able to reach servers in your local network by using their names, you will need additional exceptions that look like this:

```
forward           localhost/     .
```

7.5.3. Advanced Forwarding Examples

If you have links to multiple ISPs that provide various special content only to their subscribers, you can configure multiple Privoxies which have connections to the respective ISPs to act as forwarders to each other, so that *your* users can see the internal content of all ISPs.

Assume that host-a has a PPP connection to isp-a.example.net. And host-b has a PPP connection to isp-b.example.org. Both run Privoxy. Their forwarding configuration can look like this:

host-a:

```
forward    /           .
forward    .isp-b.example.net  host-b:8118
```

host-b:

```
forward    /           .
forward    .isp-a.example.org  host-a:8118
```

Now, your users can set their browser's proxy to use either host-a or host-b and be able to browse the internal content of both isp-a and isp-b.

If you intend to chain Privoxy and squid locally, then chaining as `browser -> squid -> privoxy` is the recommended way.

Assuming that Privoxy and squid run on the same box, your squid configuration could then look like this:

```
# Define Privoxy as parent proxy (without ICP)
cache_peer 127.0.0.1 parent 8118 7 no-query

# Define ACL for protocol FTP
acl ftp proto FTP

# Do not forward FTP requests to Privoxy
always_direct allow ftp

# Forward all the rest to Privoxy
never_direct allow all
```

You would then need to change your browser's proxy settings to squid's address and port. Squid normally uses port 3128. If unsure consult `http_port` in `squid.conf`.

You could just as well decide to only forward requests you suspect of leading to Windows executables through a virus-scanning parent proxy, say, on `antivir.example.com`, port 8010:

```
forward   /                       .
forward   /.*\.(exe|com|dll|zip)$    antivir.example.com:8010
```

7.5.4. forwarded-connect-retries

Specifies:

How often Privoxy retries if a forwarded connection request fails.

Type of value:

Number of retries.

Default value:

0

Effect if unset:

Connections forwarded through other proxies are treated like direct connections and no retry attempts are made.

Notes:

forwarded-connect-retries is mainly interesting for socks4a connections, where Privoxy can't detect why the connections failed. The connection might have failed because of a DNS timeout in which case a retry makes sense, but it might also have failed because the server doesn't exist or isn't reachable. In this case the retry will just delay the appearance of Privoxy's error message.

Note that in the context of this option, "forwarded connections" includes all connections that Privoxy forwards through other proxies. This option is not limited to the HTTP CONNECT method.

Only use this option, if you are getting lots of forwarding-related error messages that go away when you try again manually. Start with a small value and check Privoxy's logfile from time to time, to see how many retries are usually needed.

Examples:

forwarded-connect-retries 1

7.6. Miscellaneous

7.6.1. accept-intercepted-requests

Specifies:

Whether intercepted requests should be treated as valid.

Type of value:

0 or 1

Default value:

0

Effect if unset:

Only proxy requests are accepted, intercepted requests are treated as invalid.

Notes:

If you don't trust your clients and want to force them to use Privoxy, enable this option and configure your packet filter to redirect outgoing HTTP connections into Privoxy.

Note that intercepting encrypted connections (HTTPS) isn't supported.

Make sure that Privoxy's own requests aren't redirected as well. Additionally take care that Privoxy can't intentionally connect to itself, otherwise you could run into redirection loops if Privoxy's listening port is reachable by the outside or an attacker has access to the pages you visit.

If you are running Privoxy as intercepting proxy without being able to intercept all client requests you may want to adjust the CGI templates to make sure they don't reference content from config.privoxy.org.

Examples:

accept-intercepted-requests 1

7.6.2. allow-cgi-request-crunching

Specifies:

Whether requests to Privoxy's CGI pages can be blocked or redirected.

Type of value:

0 or 1

Default value:

0

Effect if unset:

Privoxy ignores block and redirect actions for its CGI pages.

Notes:

By default Privoxy ignores block or redirect actions for its CGI pages. Intercepting these requests can be useful in multi-user setups to implement fine-grained access control, but it can also render the complete web interface useless and make debugging problems painful if done without care.

Don't enable this option unless you're sure that you really need it.

Examples:

allow-cgi-request-crunching 1

7.6.3. split-large-forms

Specifies:

Whether the CGI interface should stay compatible with broken HTTP clients.

Type of value:

0 or 1

Default value:

0

Effect if unset:

The CGI form generate long GET URLs.

Notes:

Privoxy's CGI forms can lead to rather long URLs. This isn't a problem as far as the HTTP standard is concerned, but it can confuse clients with arbitrary URL length limitations.

Enabling split-large-forms causes Privoxy to divide big forms into smaller ones to keep the URL length down. It makes editing a lot less convenient and you can no longer submit all changes at once, but at least it works around this browser bug.

If you don't notice any editing problems, there is no reason to enable this option, but if one of the submit buttons appears to be broken, you should give it a try.

Examples:

split-large-forms 1

7.6.4. keep-alive-timeout

Specifies:

Number of seconds after which an open connection will no longer be reused.

Type of value:

Time in seconds.

Default value:

None

Effect if unset:

Connections are not kept alive.

Notes:

This option allows clients to keep the connection to Privoxy alive. If the server supports it, Privoxy will keep the connection to the server alive as well. Under certain circumstances this may result in speed-ups.

By default, Privoxy will close the connection to the server if the client connection gets closed, or if the specified timeout has been reached without a new request coming in. This behaviour can be changed with the connection-sharing option.

This option has no effect if Privoxy has been compiled without keep-alive support.

Note that a timeout of five seconds as used in the default configuration file significantly decreases the number of connections that will be reused. The value is used because some browsers limit the number of connections they open to a single host and apply the same limit to proxies. This can result in a single website "grabbing" all the connections the browser allows, which means connections to other websites can't be opened until the connections currently in use time out.

Several users have reported this as a Privoxy bug, so the default value has been reduced. Consider increasing it to 300 seconds or even more if you think your browser can handle it. If your browser appears to be hanging, it probably can't.

Examples:

keep-alive-timeout 300

7.6.5. tolerate-pipelining

Specifies:

Whether or not pipelined requests should be served.

Type of value:

0 or 1.

Default value:

None

Effect if unset:

If Privoxy receives more than one request at once, it terminates the client connection after serving the first one.

Notes:

Privoxy currently doesn't pipeline outgoing requests, thus allowing pipelining on the client connection is not guaranteed to improve the performance.

By default Privoxy tries to discourage clients from pipelining by discarding aggressively pipelined requests, which forces the client to resend them through a new connection.

This option lets Privoxy tolerate pipelining. Whether or not that improves performance mainly depends on the client configuration.

If you are seeing problems with pages not properly loading, disabling this option could work around the problem.

Examples:

tolerate-pipelining 1

7.6.6. default-server-timeout

Specifies:

Assumed server-side keep-alive timeout if not specified by the server.

Type of value:

Time in seconds.

Default value:

None

Effect if unset:

Connections for which the server didn't specify the keep-alive timeout are not reused.

Notes:

Enabling this option significantly increases the number of connections that are reused, provided the keep-alive-timeout option is also enabled.

While it also increases the number of connections problems when Privoxy tries to reuse a connection that already has been closed on the server side, or is closed while Privoxy is trying to reuse it, this should only be a problem if it happens for the first request sent by the client. If it happens for requests on reused client connections, Privoxy will simply close the connection and the client is supposed to retry the request without bothering the user.

Enabling this option is therefore only recommended if the connection-sharing option is disabled.

It is an error to specify a value larger than the keep-alive-timeout value.

This option has no effect if Privoxy has been compiled without keep-alive support.

Examples:

default-server-timeout 60

7.6.7. connection-sharing

Specifies:

Whether or not outgoing connections that have been kept alive should be shared between different incoming connections.

Type of value:

0 or 1

Default value:

None

Effect if unset:

Connections are not shared.

Notes:

This option has no effect if Privoxy has been compiled without keep-alive support, or if it's disabled.

Notes:

Note that reusing connections doesn't necessary cause speedups. There are also a few privacy implications you should be aware of.

If this option is effective, outgoing connections are shared between clients (if there are more than one) and closing the browser that initiated the outgoing connection does no longer affect the connection between Privoxy and the server unless the client's request hasn't been completed yet.

If the outgoing connection is idle, it will not be closed until either Privoxy's or the server's timeout is reached. While it's open, the server knows that the system running Privoxy is still there.

If there are more than one client (maybe even belonging to multiple users), they will be able to reuse each others connections. This is potentially dangerous in case of authentication schemes like NTLM where only the connection is authenticated, instead of requiring authentication for each request.

If there is only a single client, and if said client can keep connections alive on its own, enabling this option has next to no effect. If the client doesn't support connection keep-alive, enabling this option may make sense as it allows Privoxy to keep outgoing connections alive even if the client itself doesn't support it.

You should also be aware that enabling this option increases the likelihood of getting the "No server or forwarder data" error message, especially if you are using a slow connection to the Internet.

This option should only be used by experienced users who understand the risks and can weight them against the benefits.

Examples:

connection-sharing 1

7.6.8. socket-timeout

Specifies:

Number of seconds after which a socket times out if no data is received.

Type of value:

Time in seconds.

Default value:

None

Effect if unset:

A default value of 300 seconds is used.

Notes:

The default is quite high and you probably want to reduce it. If you aren't using an occasionally slow proxy like Tor, reducing it to a few seconds should be fine.

Examples:

socket-timeout 300

7.6.9. max-client-connections

Specifies:

Maximum number of client connections that will be served.

Type of value:

Positive number.

Default value:

128

Effect if unset:

Connections are served until a resource limit is reached.

Notes:

Privoxy creates one thread (or process) for every incoming client connection that isn't rejected based on the access control settings.

If the system is powerful enough, Privoxy can theoretically deal with several hundred (or thousand) connections at the same time, but some operating systems enforce resource limits by shutting down offending processes and their default limits may be below the ones Privoxy would require under heavy load.

Configuring Privoxy to enforce a connection limit below the thread or process limit used by the operating system makes sure this doesn't happen. Simply increasing the operating system's limit would work too, but if Privoxy isn't the only application running on the system, you may actually want to limit the resources used by Privoxy.

If Privoxy is only used by a single trusted user, limiting the number of client connections is probably unnecessary. If there are multiple possibly untrusted users you probably still want to additionally use a packet filter to limit the maximal number of incoming connections per client. Otherwise a malicious user could intentionally create a high number of connections to prevent other users from using Privoxy.

Obviously using this option only makes sense if you choose a limit below the one enforced by the operating system.

One most POSIX-compliant systems Privoxy can't properly deal with more than FD_SETSIZE file descriptors at the same time and has to reject connections if the limit is reached. This will likely change in a future version, but currently this limit can't be increased without recompiling Privoxy with a different FD_SETSIZE limit.

Examples:

max-client-connections 256

7.6.10. handle-as-empty-doc-returns-ok

Specifies:

The status code Privoxy returns for pages blocked with `+handle-as-empty-document`.

Type of value:

0 or 1

Default value:

0

Effect if unset:

Privoxy returns a status 403(forbidden) for all blocked pages.

Effect if set:

Privoxy returns a status 200(OK) for pages blocked with +handle-as-empty-document and a status 403(Forbidden) for all other blocked pages.

Notes:

This directive was added as a work-around for Firefox bug 492459: "Websites are no longer rendered if SSL requests for JavaScripts are blocked by a proxy." (https://bugzilla.mozilla.org/show_bug.cgi?id=492459), the bug has been fixed for quite some time, but this directive is also useful to make it harder for websites to detect whether or not resources are being blocked.

7.6.11. enable-compression

Specifies:

Whether or not buffered content is compressed before delivery.

Type of value:

0 or 1

Default value:

0

Effect if unset:

Privoxy does not compress buffered content.

Effect if set:

Privoxy compresses buffered content before delivering it to the client, provided the client supports it.

Notes:

This directive is only supported if Privoxy has been compiled with FEATURE_COMPRESSION, which should not to be confused with FEATURE_ZLIB.

Compressing buffered content is mainly useful if Privoxy and the client are running on different systems. If they are running on the same system, enabling compression is likely to slow things down. If you didn't measure otherwise, you should assume that it does and keep this option disabled.

Privoxy will not compress buffered content below a certain length.

7.6.12. compression-level

Specifies:

The compression level that is passed to the zlib library when compressing buffered content.

Type of value:

Positive number ranging from 0 to 9.

Default value:

1

Notes:

Compressing the data more takes usually longer than compressing it less or not compressing it at all. Which level is best depends on the connection between Privoxy and the client. If you can't be bothered to benchmark it for yourself, you should stick with the default and keep compression disabled.

If compression is disabled, the compression level is irrelevant.

Examples:

```
# Best speed (compared to the other levels)
compression-level 1
# Best compression
compression-level 9
# No compression. Only useful for testing as the added header
# slightly increases the amount of data that has to be sent.
# If your benchmark shows that using this compression level
# is superior to using no compression at all, the benchmark
# is likely to be flawed.
compression-level 0
```

7.6.13. client-header-order

Specifies:

The order in which client headers are sorted before forwarding them.

Type of value:

Client header names delimited by spaces or tabs

Default value:

None

Notes:

By default Privoxy leaves the client headers in the order they were sent by the client. Headers are modified in-place, new headers are added at the end of the already existing headers.

The header order can be used to fingerprint client requests independently of other headers like the User-Agent.

This directive allows to sort the headers differently to better mimic a different User-Agent.

Client headers will be emitted in the order given, headers whose name isn't explicitly specified are added at the end.

Note that sorting headers in an uncommon way will make fingerprinting actually easier. Encrypted headers are not affected by this directive.

7.6.14. client-specific-tag

Specifies:

The name of a tag that will always be set for clients that requested it through the webinterface.

Type of value:

Tag name followed by a description that will be shown in the webinterface

Default value:

None

Notes:

Warning
This is an experimental feature. The syntax is likely to change in future versions.

Client-specific tags allow Privoxy admins to create different profiles and let the users chose which one they want without impacting other users.

One use case is allowing users to circumvent certain blocks without having to allow them to circumvent all blocks. This is not possible with the enable-remote-toggle feature because it would bluntly disable all blocks for all users and also affect other actions like filters. It also is set globally which renders it useless in most multi-user setups.

After a client-specific tag has been defined with the client-specific-tag directive, action sections can be activated based on the tag by using a CLIENT-TAG pattern. The CLIENT-TAG pattern is evaluated at the same priority as URL patterns, as a result the last matching pattern wins. Tags that are created based on client or server headers are evaluated later on and can overrule CLIENT-TAG and URL patterns!

The tag is set for all requests that come from clients that requested it to be set. Note that "clients" are differentiated by IP address, if the IP address changes the tag has to be requested again.

Clients can request tags to be set by using the CGI interface http://config.privoxy.org/client-tags. The specific tag description is only used on the web page and should be phrased in away that the user understand the effect of the tag.

Examples:

```
# Define a couple of tags, the described effect requires action sections
# that are enabled based on CLIENT-TAG patterns.
client-specific-tag circumvent-blocks Overrule blocks but do not affect other actions
disable-content-filters Disable content-filters but do not affect other actions
```

7.6.15. client-tag-lifetime

Specifies:

How long a temporarily enabled tag remains enabled.

Type of value:

Time in seconds.

Default value:

60

Notes:

Warning
This is an experimental feature. The syntax is likely to change in future versions.

In case of some tags users may not want to enable them permanently, but only for a short amount of time, for example to circumvent a block that is the result of an overly-broad URL pattern.

The CGI interface http://config.privoxy.org/client-tags therefore provides a "enable this tag temporarily" option. If it is used, the tag will be set until the client-tag-lifetime is over.

Examples:

```
# Increase the time to life for temporarily enabled tags to 3 minutes
client-tag-lifetime 180
```

7.6.16. trust-x-forwarded-for

Specifies:

Whether or not Privoxy should use IP addresses specified with the X-Forwarded-For header

Type of value:

0 or one

Default value:

0

Notes:

Warning
This is an experimental feature. The syntax is likely to change in future versions.

If clients reach Privoxy through another proxy, for example a load balancer, Privoxy can't tell the client's IP address from the connection. If multiple clients use the same proxy, they will share the same client tag settings which is usually not desired.

This option lets Privoxy use the X-Forwarded-For header value as client IP address. If the proxy sets the header, multiple clients using the same proxy do not share the same client tag settings.

This option should only be enabled if Privoxy can only be reached through a proxy and if the proxy can be trusted to set the header correctly. It is recommended that ACL are used to

make sure only trusted systems can reach Privoxy.

If access to Privoxy isn't limited to trusted systems, this option would allow malicious clients to change the client tags for other clients or increase Privoxy's memory requirements by registering lots of client tag settings for clients that don't exist.

Examples:

```
# Allow systems that can reach Privoxy to provide the client
# IP address with a X-Forwarded-For header.
trust-x-forwarded-for 1
```

7.7. Windows GUI Options

Privoxy has a number of options specific to the Windows GUI interface:

If "activity-animation" is set to 1, the Privoxy icon will animate when "Privoxy" is active. To turn off, set to 0.

```
activity-animation 1
```

If "log-messages" is set to 1, Privoxy copies log messages to the console window. The log detail depends on the debug directive.

```
log-messages 1
```

If "log-buffer-size" is set to 1, the size of the log buffer, i.e. the amount of memory used for the log messages displayed in the console window, will be limited to "log-max-lines" (see below).

Warning: Setting this to 0 will result in the buffer to grow infinitely and eat up all your memory!

```
log-buffer-size 1
```

log-max-lines is the maximum number of lines held in the log buffer. See above.

```
log-max-lines 200
```

If "log-highlight-messages" is set to 1, Privoxy will highlight portions of the log messages with a bold-faced font:

```
log-highlight-messages 1
```

The font used in the console window:

```
log-font-name Comic Sans MS
```

Font size used in the console window:

```
log-font-size 8
```

"show-on-task-bar" controls whether or not Privoxy will appear as a button on the Task bar when minimized:

```
show-on-task-bar 0
```

If "close-button-minimizes" is set to 1, the Windows close button will minimize Privoxy instead of closing the program (close with the exit option on the File menu).

```
close-button-minimizes 1
```

The "hide-console" option is specific to the MS-Win console version of Privoxy. If this option is used, Privoxy will disconnect from and hide the command console.

```
#hide-console
```

Prev Home Next
Privoxy Configuration Actions Files