Privoxy User Manual

By: Privoxy Developers

The user manual gives users information on how to install, configure and use Privoxy (http://www.privoxy.org/).

Privoxy is a web proxy with advanced filtering capabilities for protecting privacy, filtering web page content, managing cookies, controlling access, and removing ads, banners, pop-ups and other obnoxious Internet junk. Privoxy has a very flexible configuration and can be customized to suit individual needs and tastes. Privoxy has application for both stand-alone systems and multi-user networks.

Privoxy is based on Internet Junkbuster (tm).

You can find the latest version of the user manual at http://www.privoxy.org/user-manual/. Please see the Contact section (contact.html) on how to contact the developers.

1. Introduction

This documentation is included with the current beta version of Privoxy, v.2.9.14, and is mostly complete at this point. The most up to date reference for the time being is still the comments in the source files and in the individual configuration files. Development of version 3.0 is currently nearing completion, and includes many significant changes and enhancements over earlier versions. The target release date for stable v3.0 is “soon” ;-).

Since this is a beta version, not all new features are well tested. This documentation may be slightly out of sync as a result (especially with CVS sources). And there *may be* bugs, though hopefully not many!

Features

In addition to Internet Junkbuster’s traditional features of ad and banner blocking and cookie management, Privoxy provides new features, some of them currently under development:

- FIXME: complete the list of features. change the order: most important features to the top of the list. prefix new features with "NEW".
- Integrated browser based configuration and control utility at http://config.privoxy.org/ (shortcut: http://p.p/). Browser-based tracing of rule and filter effects. Remote toggling.
- Blocking of annoying pop-up browser windows.
- HTTP/1.1 compliant (but not all optional 1.1 features are supported).
- Support for Perl Compatible Regular Expressions in the configuration files, and generally a more sophisticated and flexible configuration syntax over previous versions.
- GIF de-animation.
- Web page content filtering (removes banners based on size, invisible “web-bugs”, JavaScript and HTML annoyances, pop-ups, etc.)
- Bypass many click-tracking scripts (avoids script redirection).
- Multi-threaded (POSIX and native threads).
- Auto-detection and re-reading of config file changes.
- User-customizable HTML templates (e.g. 404 error page).
- Improved cookie management features (e.g. session based cookies).
- Improved signal handling, and a true daemon mode (Unix).
- Every feature now controllable on a per-site or per-location basis, configuration more powerful and versatile over-all.
- Many smaller new features added, limitations and bugs removed, and security holes fixed.

Installation

Privoxy is available both in convenient pre-compiled packages for a wide range of operating systems, and as raw source code. For most users, we recommend using the packages, which can be downloaded from our Privoxy Project Page (http://sourceforge.net/projects/ijbswa/). For installing and compiling the source code, please look into our Developer Manual.

If you like to live on the bleeding edge and are not afraid of using possibly unstable development versions, you can check out the up-to-the-minute version directly from the CVS repository (http://sourceforge.net/cvs/?group_id=11118) or simply download the nightly CVS tarball.

(http://cvs.sourceforge.net/cvstarballs/ijbswa-cvsroot.tar.gz) Again, we refer you to the Developer Manual.

At present, Privoxy is known to run on Windows(95, 98, ME, 2000, XP), Linux (RedHat, Suse, Debian), Mac OSX, OS/2, AmigaOS, FreeBSD, NetBSD, BeOS, and many more flavors of Unix.

Note: If you have a previous Junkbuster or Privoxy installation on your system, you will need to remove it. Some platforms do this for you as part of their installation procedure. (See below for your platform).

In any case *be sure to backup your old configuration if it is valuable to you.* See the note to upgraders section below.

Red Hat and SuSE RPMs

RPMs can be installed with `rpm -Uvh privoxy-2.9.14-1.rpm`, and will use `/etc/privoxy` for the location of configuration files.

Note that on Red Hat, Privoxy will *not* be automatically started on system boot. You will need to enable that using **chkconfig**, **ntsysv**, or similar methods. Note that SuSE will automatically start Privoxy in the boot process.

If you have problems with failed dependencies, try rebuilding the SRC RPM: `rpm --rebuild privoxy-2.9.14-1.src.rpm;`. This will use your locally installed libraries and RPM version.

Also note that if you have a Junkbuster RPM installed on your system, you need to remove it first, because the packages conflict. Otherwise, RPM will try to remove Junkbuster automatically, before installing Privoxy.

Debian

FIXME.

Windows

Just double-click the installer, which will guide you through the installation process. You will find the configuration files in the same directory as you installed Privoxy in. We do not use the registry of Windows.

Solaris, NetBSD, FreeBSD, HP-UX

Create a new directory, `cd` to it, then unzip and untar the archive. For the most part, you'll have to figure out where things go. FIXME.

OS/2

First, make sure that no previous installations of Junkbuster and / or Privoxy are left on your system. You can do this by

Then, just double-click the WarpIN self-installing archive, which will guide you through the installation process. A shadow of the Privoxy executable will be placed in your startup folder so it will start automatically whenever OS/2 starts.

The directory you choose to install Privoxy into will contain all of the configuration files.

Max OSX

Unzip the downloaded package (you can either double-click on the file in the finder, or on the desktop if you downloaded it there). Then, double-click on the package installer icon and follow the installation process. Privoxy will be installed in the subdirectory `/Applications/Privoxy.app`. Privoxy will set itself up to start automatically on system bring-up via `/System/Library/StartupItems/Privoxy`.

AmigaOS

Copy and then unpack the `lha` archive to a suitable location. All necessary files will be installed into Privoxy directory, including all configuration and log files. To uninstall, just remove this directory.

Start Privoxy (with RUN <>NIL:) in your `startnet` script (AmiTCP), in `s:user-startup` (RoadShow), as startup program in your startup script (Genesis), or as startup action (Miami and MiamiDx). Privoxy will automatically quit when you quit your TCP/IP stack (just ignore the harmless warning your TCP/IP stack may display that Privoxy is still running).

Note to Upgraders

There are very significant changes from older versions of Junkbuster to the current Privoxy. Configuration is substantially changed. Junkbuster 2.0.x and earlier configuration files will not migrate.

The functionality of the old `blockfile`, `cookiefile` and `imagelist`, are now combined into the "actions files". `default.action`, is the main actions file. Local exceptions should best be put into `user.action`.

A "filter file" (typically `default.filter`) is new as of Privoxy 2.9.x, and provides some of the new sophistication (explained below). `config` is much the same as before.

If upgrading from a 2.0.x version, you will have to use the new config files, and possibly adapt any personal rules from your older files. When porting personal rules over from the old `blockfile` to the new actions files, please note that even the pattern syntax has changed. If upgrading from 2.9.x development versions, it is still recommended to use the new configuration files.

A quick list of things to be aware of before upgrading:

- The default listening port is now 8118 due to a conflict with another service (NAS).
- Some installers may remove earlier versions completely. Save any important configuration files!
- Privoxy is controllable with a web browser at the special URL: http://config.privoxy.org/ (Shortcut: http://p.p/). Many aspects of configuration can be done here, including temporarily disabling Privoxy.
- The primary configuration file for cookie management, ad and banner blocking, and many other aspects of Privoxy configuration is in the "actions" files. It is strongly recommended to become familiar with the new actions concept below, before modifying these files. Locally defined rules should go into `user.action`.
- Some installers may not automatically start Privoxy after installation.

Quickstart to Using Privoxy

- Install Privoxy. See the section Installing.
- Start Privoxy. See the section Starting Privoxy.
- Change your browser's configuration to use the proxy `localhost` on port `8118`. See the section Starting Privoxy.
- Enjoy surfing with enhanced comfort and privacy. Please see the section Contacting the Developers on how to report bugs or problems with websites or to get help. You may want to change the file `user.action` to further tweak your new browsing experience.

Starting Privoxy

Before launching Privoxy for the first time, you will want to configure your browser(s) to use Privoxy as a HTTP and HTTPS proxy. The default is localhost for the proxy address, and port 8118 (earlier versions used port 8000). This is the one configuration step that must be done!

With Netscape (and Mozilla), this can be set under `Edit -> Preferences -> Advanced -> Proxies -> HTTP Proxy`. For Internet Explorer: `Tools -> Internet Properties -> Connections -> LAN Setting`. Then, check "Use Proxy" and fill in the appropriate info (Address: localhost, Port: 8118). Include if HTTPS proxy support too.

After doing this, flush your browser's disk and memory caches to force a re-reading of all pages and to get rid of any ads that may be cached. You are now ready to start enjoying the benefits of using Privoxy!

Privoxy is typically started by specifying the main configuration file to be used on the command line. Example Unix startup command:

```
# /usr/sbin/privoxy /etc/privoxy/config
```

See below for other command line options.

An init script is provided for SuSE and Red Hat.

For for SuSE: **rcprivoxy start**

For Red Hat and Debian: **/etc/rc.d/init.d/privoxy start**

If no configuration file is specified on the command line, Privoxy will look for a file named `config` in the current directory. Except on Win32 where it will try `config.txt`. If no file is specified on the command line and no default configuration file can be found, Privoxy will fail to start.

The included default configuration files should give a reasonable starting point. Most of the per site configuration is done in the "actions" files. These are where various cookie actions are defined, ad and banner blocking, and other aspects of Privoxy configuration. There are several such files included, with varying levels of aggressiveness.

You will probably want to keep an eye out for sites for which you may prefer persistent cookies, and add these to your actions configuration as needed. By default, most of these will be accepted only during the current browser session (aka "session cookies"), unless you add them to the configuration. If you want the browser to handle this instead, you will need to edit `user.action` (or through the web based interface) and disable this feature. If you use more than one browser, it would make more sense to let Privoxy handle this. In which case, the browser(s) should be set to accept all cookies.

Another feature where you will probably want to define exceptions for trusted sites is the popup-killing (through the `+popup` and `+filter{popups}` actions), because your favorite shopping, banking, or

leisure site may need popups (explained below).

Privoxy is HTTP/1.1 compliant, but not all of the optional 1.1 features are as yet supported. In the unlikely event that you experience inexplicable problems with browsers that use HTTP/1.1 per default (like Mozilla or recent versions of I.E.), you might try to force HTTP/1.0 compatibility. For Mozilla, look under `Edit -> Preferences -> Debug -> Networking`. Alternatively, set the "+downgrade-http-version" config option in `default.action` which will downgrade your browser's HTTP requests from HTTP/1.1 to HTTP/1.0 before processing them.

After running Privoxy for a while, you can start to fine tune the configuration to suit your personal, or site, preferences and requirements. There are many, many aspects that can be customized. "Actions" can be adjusted by pointing your browser to http://config.privoxy.org/ (shortcut: http://p.p/), and then follow the link to "View & Change the Current Configuration". (This is an internal page and does not require Internet access.)

In fact, various aspects of Privoxy configuration can be viewed from this page, including current configuration parameters, source code version numbers, the browser's request headers, and "actions" that apply to a given URL. In addition to the actions file editor mentioned above, Privoxy can also be turned "on" and "off" (toggled) from this page.

If you encounter problems, try loading the page without Privoxy. If that helps, enter the URL where you have the problems into the browser based rule tracing utility (http://p.p/show-url-info). See which rules apply and why, and then try turning them off for that site one after the other, until the problem is gone. When you have found the culprit, you might want to turn the rest on again.

If the above paragraph sounds gibberish to you, you might want to read more about the actions concept (configuration.html#ACTIONSFILE) or even dive deep into the Appendix on actions (appendix.html#ACTIONSANAT).

If you can't get rid of the problem at all, think you've found a bug in Privoxy, want to propose a new feature or smarter rules, please see the section "Contacting the Developers" (contact.html) below.

Command Line Options

Privoxy may be invoked with the following command-line options:

- *--version*

  Print version info and exit. Unix only.

- *--help*

  Print short usage info and exit. Unix only.

- *--no-daemon*

  Don’t become a daemon, i.e. don’t fork and become process group leader, and don’t detach from controlling tty. Unix only.

- *--pidfile FILE*

  On startup, write the process ID to *FILE*. Delete the *FILE* on exit. Failure to create or delete the *FILE* is non-fatal. If no *FILE* option is given, no PID file will be used. Unix only.

- *--user USER[.GROUP]*

  After (optionally) writing the PID file, assume the user ID of *USER*, and if included the GID of GROUP. Exit if the privileges are not sufficient to do so. Unix only.

- *configfile*

  If no *configfile* is included on the command line, Privoxy will look for a file named “config” in the current directory (except on Win32 where it will look for “config.txt” instead). Specify full path to avoid confusion. If no config file is found, Privoxy will fail to start.

Privoxy Configuration

All Privoxy configuration is stored in text files. These files can be edited with a text editor. Many important aspects of Privoxy can also be controlled easily with a web browser.

Controlling Privoxy with Your Web Browser

Privoxy’s user interface can be reached through the special URL http://config.privoxy.org/ (shortcut: http://p.p/), which is a built-in page and works without Internet access. You will see the following section:

Privoxy Menu

- View & change the current configuration (http://config.privoxy.org/show-status
- View the source code version numbers (http://config.privoxy.org/show-version)
- View the request headers. (http://config.privoxy.org/show-request)
- Look up which actions apply to a URL and why (http://config.privoxy.org/show-u
- Toggle Privoxy on or off (http://config.privoxy.org/toggle)

This should be self-explanatory. Note the first item leads to an editor for the "actions list", which is where the ad, banner, cookie, and URL blocking magic is configured as well as other advanced features of Privoxy. This is an easy way to adjust various aspects of Privoxy configuration. The actions file, and other configuration files, are explained in detail below.

"Toggle Privoxy On or Off" is handy for sites that might have problems with your current actions and filters. You can in fact use it as a test to see whether it is Privoxy causing the problem or not. Privoxy continues to run as a proxy in this case, but all filtering is disabled. There is even a toggle Bookmarklet offered, so that you can toggle Privoxy with one click from your browser.

Configuration Files Overview

For Unix, *BSD and Linux, all configuration files are located in `/etc/privoxy/` by default. For MS Windows, OS/2, and AmigaOS these are all in the same directory as the Privoxy executable. The name and number of configuration files has changed from previous versions, and is subject to change as development progresses.

The installed defaults provide a reasonable starting point, though some settings may be aggressive by some standards. For the time being, the principle configuration files are:

- The main configuration file is named config on Linux, Unix, BSD, OS/2, and AmigaOS and `config.txt` on Windows. This is a required file.
- `default.action` (the main actions file) is used to define the default settings for various "actions" relating to images, banners, pop-ups, access restrictions, banners and cookies.

  Multiple actions files may be defined in `config`. These are processed in the order they are defined. Local customizations and locally preferred exceptions to the default policies as defined in `default.action` are probably best applied in `user.action`, which should be preserved across upgrades. `standard.action` is also included. This is mostly for Privoxy's internal use.

  There is also a web based editor that can be accessed from http://config.privoxy.org/show-status/ (Shortcut: http://p.p/show-status/) for the various actions files.

- `default.filter` (the filter file) can be used to re-write the raw page content, including viewable text as well as embedded HTML and JavaScript, and whatever else lurks on any given web page. The filtering jobs are only pre-defined here; whether to apply them or not is up to the actions files.

All files use the “#” character to denote a comment (the rest of the line will be ignored) angd understand line continuation through placing a backslash ("\") as the very last character in a line. If the # is preceded by a backslash, it looses its special function. Placing a # in front of an otherwise valid configuration line to prevent it from being interpreted is called "commenting out" that line.

The actions files and `default.filter` can use Perl style regular expressions for maximum flexibility.

After making any changes, there is no need to restart Privoxy in order for the changes to take effect. Privoxy detects such changes automatically. Note, however, that it may take one or two additional requests for the change to take effect. When changing the listening address of Privoxy, these “wake up” requests must obviously be sent to the *old* listening address.

While under development, the configuration content is subject to change. The below documentation may not be accurate by the time you read this. Also, what constitutes a “default” setting, may change, so please check all your configuration files on important issues.

The Main Configuration File

Again, the main configuration file is named `config` on Linux/Unix/BSD and OS/2, and `config.txt` on Windows. 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).

Configuration and Log File Locations

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

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

When development goes modular and multi-user, the blocker, filter, and per-user config will be stored in subdirectories of "confdir". For now, the configuration directory structure is flat, except for `confdir/templates`, where the HTML templates for CGI output reside (e.g. Privoxy's 404 error page).

logdir

Specifies:

The directory where all logging takes place (i.e. where `logfile` and `jarfile` are 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

actionsfile

Specifies:

The actions file(s) to use

Type of value:

File name, relative to `confdir`

Default value:

```
standard     # Internal purposes, recommended not editing
default      # Main actions file
user         # User customizations
```

Effect if unset:

No actions are taken at all. Simple neutral proxying.

Notes:

Multiple `actionsfile` lines are OK and are in fact recommended!

The default values include standard.action, which is used for internal purposes and should be loaded, default.action, which is the "main" actions file maintained by the developers, and `user.action`, where you can make your personal additions.

There is no point in using Privoxy without an actions file.

filterfile

Specifies:

The filter file 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 off

Notes:

The “default.filter” file contains content modification rules that use “regular expressions”. These rules permit powerful changes on the content of Web pages, e.g., you could disable your favorite JavaScript annoyances, re-write the actual displayed text, or just have some fun replacing “Microsoft” with “MicroSuck” wherever it appears on a Web page.

logfile

Specifies:

The log file to use

Type of value:

File name, relative to `logdir`

Default value:

logfile (Unix) *or* privoxy.log (Windows)

Effect if unset:

No log file is used, all log messages go to the console (`stderr`).

Notes:

The windows version will additionally log to the console.

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) but in most cases you probably will never look at it.

Your logfile will grow indefinitely, and you will probably want to periodically remove it. On Unix systems, you can do this with a cron job (see “man cron”). For Red Hat, a **logrotate** script has been included.

On SuSE Linux systems, you can place a line like “/var/log/privoxy.* +1024k 644 nobody.nogroup” in `/etc/logfiles`, with the effect that cron.daily will automatically archive, gzip, and empty the log, when it exceeds 1M size.

jarfile

Specifies:

The file to store intercepted cookies in

Type of value:

File name, relative to `logdir`

Default value:

jarfile (Unix) *or* privoxy.jar (Windows)

Effect if unset:

Intercepted cookies are not stored at all.

Notes:

The jarfile may grow to ridiculous sizes over time.

trustfile

Specifies:

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 whole trust mechanism is turned off.

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 named in the trustfile. You can also mark sites as trusted referrers (with +), with the effect that access to untrusted sites will be granted, if a link from a trusted referrer was used. The link target will then be added to the “trustfile”. Possible applications include limiting Internet access for children.

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

Local Set-up Documentation

If you intend to operate Privoxy for more users that 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.

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:

Two example URL are provided

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` above.)

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!

admin-address

Specifies:

An email address to reach the proxy 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.

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 ;-)

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.

debug

Specifies:

Key values that determine what information gets logged.

Type of value:

Integer values

Default value:

12289 (i.e.: URLs plus informational and warning messages)

Effect if unset:

Nothing gets logged.

Notes:

The available debug levels are:

```
debug         1 # show each GET/POST/CONNECT request
debug         2 # show each connection status
debug         4 # show I/O status
debug         8 # show header parsing
debug        16 # log all data into the logfile
debug        32 # debug force feature
```

```
debug        64 # debug regular expression filter
debug       128 # debug fast redirects
debug       256 # debug GIF de-animation
debug       512 # Common Log Format
debug      1024 # debug kill pop-ups
debug      4096 # Startup banner and warnings.
debug      8192 # Non-fatal errors
```

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, 4096 and 8192 are highly 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).

The reporting of *fatal* errors (i.e. ones which crash Privoxy) is always on and cannot be disabled.

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

single-threaded

Specifies:

Whether to run only one server thread

Type of value:

None

Default value:

Unset

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 debug purposes and you should never need to use it. *It will drastically reduce performance.*

Access Control and Security

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

listen-address

Specifies:

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

Type of value:

[*IP-Address*]:*Port*

Default value:

localhost:8118

Effect if unset:

Bind to localhost (127.0.0.1), 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.

If you leave out the IP address, Privoxy will bind to all interfaces (addresses) on your machine and may become reachable from the Internet. In that case, consider using access control lists (ACL's) (see "ACLs" below), or a firewall.

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
```

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. behave like a normal, content-neutral proxy. See `enable-remote-toggle` below. This is not really useful anymore, since toggling is much easier via the web interface (http://config.privoxy.org/toggle) then via editing the `conf` file.

The windows version will only display the toggle icon in the system tray if this option is present.

enable-remote-toggle

Specifies:

Whether or not the web-based toggle feature (http://config.privoxy.org/toggle) may be used

Type of value:

0 or 1

Default value:

1

Effect if unset:

The web-based toggle feature is disabled.

Notes:

When toggled off, Privoxy acts like a normal, content-neutral proxy, i.e. it acts as if none of the actions applied to any URL.

For the time being, 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 you must have compiled Privoxy with support for this feature, otherwise this option has no effect.

enable-edit-actions

Specifies:

Whether or not the web-based actions file editor (http://config.privoxy.org/show-status) may be used

Type of value:

0 or 1

Default value:

1

Effect if unset:

The web-based actions file editor is disabled.

Notes:

For the time being, 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. So this option is *not recommended* for multi-user environments with untrusted users.

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

ACLs: permit-access and deny-access

Specifies:

Who can access what.

Type of value:

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

Where *src_addr* and *dst_addr* are IP addresses in dotted decimal notation or valid DNS names, 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.

Default value:

Unset

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 or internal (home) network address by means of the `listen-address` option.

Please see the warnings in the FAQ that this proxy 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, then the Privoxy talks only 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.

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.

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:

```
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 www.dirty-stuff.example.com:

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

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.

Forwarding

This feature allows routing of HTTP requests through a chain of multiple proxies. It can be used to better protect privacy and confidentiality when accessing specific domains by routing requests to those domains through an anonymous public proxy (see e.g. http://www.multiproxy.org/anon_list.htm) Or to use a caching proxy to speed up browsing. Or chaining to a parent proxy may be necessary because the machine that Privoxy runs on has no direct Internet access.

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

forward

Specifies:

To which parent HTTP proxy specific requests should be routed.

Type of value:

`target_domain`[:*`port`*] *`http_parent`*[/*`port`*]

Where *`target_domain`* is a domain name pattern (see the chapter on domain matching in the `default.action` file), *`http_parent`* is the address of the parent HTTP proxy as an IP addresses in dotted decimal notation or as a valid DNS name (or "." to denote "no forwarding", and the optional *`port`* parameters are TCP ports, i.e. integer values from 1 to 64535

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.

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

Examples:

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

```
forward   .*     anon-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.example-isp.net:8000
forward   .example-isp.net   .
```

forward-socks4 and forward-socks4a

Specifies:

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

Type of value:

`target_domain`[:*`port`*] *`socks_proxy`*[/*`port`*] *`http_parent`*[/*`port`*]

Where *`target_domain`* is a domain name pattern (see the chapter on domain matching in the `default.action` file), *`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 64535

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.

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.example-
isp.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  .
```

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

host-a:

```
forward    .*.         .
forward    .isp-b.net  host-b:8118
```

host-b:

```
forward    .*.         .
forward    .isp-a.net  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 chain 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`.

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 will log messages to the console window:

```
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
```

Actions Files

The actions files are used to define what actions Privoxy takes for which URLs, and thus determines how ad images, cookies and various other aspects of HTTP content and transactions are handled, and on which sites (or even parts thereof). There are three such files included with Privoxy, with slightly different purposes. `default.action` sets the default policies. `standard.action` is used by Privoxy and the web based editor to set pre-defined values (and normally should not be edited). Local exceptions are best done in `user.action`. The content of these can all be viewed and edited from http://config.privoxy.org/show-status.

Anything you want can blocked, including ads, banners, or just some obnoxious URL that you would rather not see is done here. Cookies can be accepted or rejected, or accepted only during the current browser session (i.e. not written to disk), content can be modified, JavaScripts tamed, user-tracking fooled, and much more. See below for a complete list of available actions.

An actions file typically has sections. Near the top, “aliases” are optionally defined (discussed below (configuration.html#ALIASES)), then the default set of rules which will apply universally to all sites and pages. And then below that, exceptions to the defined universal policies.

Finding the Right Mix

Note that some actions like cookie suppression or script disabling may render some sites unusable, which rely on these techniques to work properly. Finding the right mix of actions is not easy and certainly a matter of personal taste. In general, it can be said that the more “aggressive” your default settings (in the top section of the actions file) are, the more exceptions for “trusted” sites you will have to make later. If, for example, you want to kill popup windows per default, you’ll have to make exceptions from that rule for sites that you regularly use and that require popups for actually useful content, like maybe your bank, favorite shop, or newspaper.

We have tried to provide you with reasonable rules to start from in the distribution actions files. But there is no general rule of thumb on these things. There just are too many variables, and sites are constantly changing. Sooner or later you will want to change the rules (and read this chapter again :).

How to Edit

The easiest way to edit the “actions” files is with a browser by using our browser-based editor, which can be reached from http://config.privoxy.org/show-status.

If you prefer plain text editing to GUIs, you can of course also directly edit the the actions files.

How Actions are Applied to URLs

Actions files are divided into sections. There are special sections, like the “alias” sections which will be discussed later. For now let’s concentrate on regular sections: They have a heading line (often split up to multiple lines for readability) which consist of a list of actions, separated by whitespace and enclosed in curly braces. Below that, there is a list of URL patterns, each on a separate line.

To determine which actions apply to a request, the URL of the request is compared to all patterns in this file. Every time it matches, the list of applicable actions for the URL is incrementally updated, using the heading of the section in which the pattern is located. If multiple matches for the same URL set the same action differently, the last match wins. If not, the effects are aggregated (e.g. a URL might match both the “+handle-as-image” (configuration.html#HANDLE-AS-IMAGE) and “+block” (configuration.html#BLOCK) actions).

You can trace this process by visiting http://config.privoxy.org/show-url-info.

More detail on this is provided in the Appendix, Anatomy of an Action.

Patterns

Generally, a pattern has the form <domain>/<path>, where both the <domain> and <path> are optional. (This is why the pattern / matches all URLs).

`www.example.com/`

is a domain-only pattern and will match any request to `www.example.com`, regardless of which document on that server is requested.

`www.example.com`

means exactly the same. For domain-only patterns, the trailing / may be omitted.

`www.example.com/index.html`

matches only the single document `/index.html` on `www.example.com`.

`/index.html`

matches the document `/index.html`, regardless of the domain, i.e. on *any* web server.

`index.html`

matches nothing, since it would be interpreted as a domain name and there is no top-level domain called `.html`.

The Domain Pattern

The matching of the domain part offers some flexible options: if the domain starts or ends with a dot, it becomes unanchored at that end. For example:

`.example.com`

matches any domain that *ENDS* in `.example.com`

`www.`

matches any domain that *STARTS* with `www.`

`.example.`

matches any domain that *CONTAINS* `.example.` (Correctly speaking: It matches any FQDN that contains `example` as a domain.)

Additionally, there are wild-cards that you can use in the domain names themselves. They work pretty similar to shell wild-cards: “*” stands for zero or more arbitrary characters, “?” stands for any single character, you can define character classes in square brackets and all of that can be freely mixed:

`ad*.example.com`

matches “adserver.example.com”, “ads.example.com”, etc but not “sfads.example.com”

`*ad*.example.com`

matches all of the above, and then some.

`.?pix.com`

matches `www.ipix.com`, `pictures.epix.com`, `a.b.c.d.e.upix.com` etc.

`www[1-9a-ez].example.c*`

matches `www1.example.com`, `www4.example.cc`, `wwwd.example.cy`, `wwwz.example.com` etc., but *not* `wwww.example.com`.

The Path Pattern

Privoxy uses Perl compatible regular expressions (through the PCRE (http://www.pcre.org/) library) for matching the path.

There is an Appendix with a brief quick-start into regular expressions, and full (very technical) documentation on PCRE regex syntax is available on-line at http://www.pcre.org/man.txt. You might also find the Perl man page on regular expressions (`man perlre`) useful, which is available on-line at http://www.perldoc.com/perl5.6/pod/perlre.html.

Note that the path pattern is automatically left-anchored at the “/”, i.e. it matches as if it would start with a “^” (regular expression speak for the beginning of a line).

Please also note that matching in the path is case *INSENSITIVE* by default, but you can switch to case sensitive at any point in the pattern by using the “(?-i)” switch: `www.example.com/(?-i)PaTtErN.*` will match only documents whose path starts with `PaTtErN` in *exactly* this capitalization.

Actions

All actions are disabled by default, until they are explicitly enabled somewhere in an actions file. Actions are turned on if preceded with a “+”, and turned off if preceded with a “-”. So a “+action” means “do that action”, e.g. “+block” means please “block the following URL patterns”.

Actions are invoked by enclosing the action name in curly braces (e.g. {+some_action}), followed by a list of URLs (or patterns that match URLs) to which the action applies. There are three classes of actions:

- Boolean, i.e the action can only be “on” or “off”. Examples:

```
{+name}        # enable this action
{-name}        # disable this action
```

- Parameterized, e.g. “+/-hide-user-agent{ Mozilla 1.0 }”, where some value is required in order to enable this type of action. Examples:

```
{+name{param}}  # enable action and set parameter to “param”
{-name}         # disable action (“parameter”) can be omitted
```

- Multi-value, e.g. “{+/-add-header{Name: value}}” or “{+/-send-wafer{name=value}}”), where some value needs to be defined in addition to simply enabling the action. Examples:

```
{+name{param=value}}   # enable action and set “param” to “value”
{-name{param=value}}   # remove the parameter “param” completely
{-name}                # disable this action totally and remove param too
```

If nothing is specified in any actions file, no “actions” are taken. So in this case Privoxy would just be a normal, non-blocking, non-anonymizing proxy. You must specifically enable the privacy and blocking features you need (although the provided default actions files will give a good starting point).

Later defined actions always over-ride earlier ones. So exceptions to any rules you make, should come in the latter part of the file. For multi-valued actions, the actions are applied in the order they are specified. Actions files are processed in the order they are defined in `config` (the default installation has three actions files). It also quite possible for any given URL pattern to match more than one action!

The list of valid Privoxy “actions” are:

+add-header{Name: value}

Type:

Multi-value.

Typical uses:

Send a user defined HTTP header to the web server.

Possible values:

Any value is possible. Validity of the defined HTTP headers is not checked.

Example usage:

```
    {+add-header{X-User-Tracking: sucks}}
.example.com
```

Notes:

This action may be specified multiple times, in order to define multiple headers. This is rarely needed for the typical user. If you don't know what "HTTP headers" are, you definitely don't need to worry about this one.

+block

Type:

Boolean.

Typical uses:

Used to block a URL from reaching your browser. The URL may be anything, but is typically used to block ads or other obnoxious content.

Possible values:

N/A

Example usage:

```
    {+block}
.banners.example.com
.ads.r.us
```

Notes:

If a URL matches one of the blocked patterns, Privoxy will intercept the URL and display its special "BLOCKED" page instead. If there is sufficient space, a large red banner will appear with a

friendly message about why the page was blocked, and a way to go there anyway. If there is insufficient space a smaller blocked page will appear without the red banner. Click here (http://ads.bannerserver.example.com/nasty-ads/sponsor.html) to view the default blocked HTML page (Privoxy must be running for this to work as intended!).

A very important exception is if the URL *matches both* "+block" and "+handle-as-image" (configuration.html#HANDLE-AS-IMAGE), then it will be handled by "+set-image-blocker" (configuration.html#SET-IMAGE-BLOCKER) (see below). It is important to understand this process, in order to understand how Privoxy is able to deal with ads and other objectionable content.

The "+filter" (configuration.html#FILTER) action can also perform some of the same functionality as "+block", but by virtue of very different programming techniques, and is most often used for different reasons.

+deanimate-gifs

Type:

Parameterized.

Typical uses:

To stop those annoying, distracting animated GIF images.

Possible values:

"last" or "first"

Example usage:

```
{+deanimate-gifs{last}}
.example.com
```

Notes:

De-animate all animated GIF images, i.e. reduce them to their last frame. This will also shrink the images considerably (in bytes, not pixels!). If the option "first" is given, the first frame of the animation is used as the replacement. If "last" is given, the last frame of the animation is used instead, which probably makes more sense for most banner animations, but also has the risk of not showing the entire last frame (if it is only a delta to an earlier frame).

+downgrade-http-version

Type:

Boolean.

Typical uses:

"+downgrade-http-version" will downgrade HTTP/1.1 client requests to HTTP/1.0 and downgrade the responses as well.

Possible values:

N/A

Example usage:

```
{+downgrade-http-version}
.example.com
```

Notes:

Use this action for servers that use HTTP/1.1 protocol features that Privoxy doesn't handle well yet. HTTP/1.1 is only partially implemented. Default is not to downgrade requests. This is an infrequently needed action, and is used to help with rare problem sites only.

+fast-redirects

Type:

Boolean.

Typical uses:

The "+fast-redirects" action enables interception of "redirect" requests from one server to another, which are used to track users.Privoxy can cut off all but the last valid URL in a redirect request and send a local redirect back to your browser without contacting the intermediate site(s).

Possible values:

N/A

Example usage:

```
{+fast-redirects}
```

.example.com

Notes:

Many sites, like yahoo.com, don't just link to other sites. Instead, they will link to some script on their own server, giving the destination as a parameter, which will then redirect you to the final target. URLs resulting from this scheme typically look like: *http://some.place/some_script?http://some.where-else*.

Sometimes, there are even multiple consecutive redirects encoded in the URL. These redirections via scripts make your web browsing more traceable, since the server from which you follow such a link can see where you go to. Apart from that, valuable bandwidth and time is wasted, while your browser ask the server for one redirect after the other. Plus, it feeds the advertisers.

This is a normally "on" feature, and often requires exceptions for sites that are sensitive to defeating this mechanism.

+filter

Type:

Parameterized.

Typical uses:

Apply page filtering as defined by named sections of the `default.filter` file to the specified site(s). "Filtering" can be any modification of the raw page content, including re-writing or deletion of content.

Possible values:

"+filter" must include the name of one of the section identifiers from `default.filter` (or whatever *filterfile* is specified in `config`).

Example usage (from the current `default.filter`):

+*filter{html-annoyances}*: Get rid of particularly annoying HTML abuse.
+*filter{js-annoyances}*: Get rid of particularly annoying JavaScript abuse
+*filter{content-cookies}*: Kill cookies that come in the HTML or JS content
+*filter{popups}*: Kill all popups in JS and HTML
+*filter{frameset-borders}*: Give frames a border and make them resizable
+*filter{webbugs}*: Squish WebBugs (1x1 invisible GIFs used for user tracking)

+*filter{refresh-tags}*: Kill automatic refresh tags (for dial-on-demand setups)
+*filter{fun}*: Text replacements for subversive browsing fun!
+*filter{nimda}*: Remove Nimda (virus) code.
+*filter{banners-by-size}*: Kill banners by size (*very* efficient!)
+*filter{shockwave-flash}*: Kill embedded Shockwave Flash objects
+*filter{crude-parental}*: Kill all web pages that contain the words "sex" or "warez"

Notes:

This is potentially a very powerful feature! And requires a knowledge of regular expressions if you want to “roll your own”. Filtering operates on a line by line basis throughout the entire page.

Filtering requires buffering the page content, which may appear to slow down page rendering since nothing is displayed until all content has passed the filters. (It does not really take longer, but seems that way since the page is not incrementally displayed.) This effect will be more noticeable on slower connections.

Filtering can achieve some of the effects as the “+block” (configuration.html#BLOCK) action, i.e. it can be used to block ads and banners. In the overall scheme of things, filtering is one of the first things “Privoxy” does with a web page. So other most other actions are applied to the already “filtered” page.

+hide-forwarded-for-headers

Type:

Boolean.

Typical uses:

Block any existing X-Forwarded-for HTTP header, and do not add a new one.

Possible values:

N/A

Example usage:

```
    {+hide-forwarded-for-headers}
.example.com
```

Notes:

It is fairly safe to leave this on. It does not seem to break many sites.

+hide-from-header

Type:

Parameterized.

Typical uses:

To block the browser from sending your email address in a “From:” header.

Possible values:

Keyword: “block”, or any user defined value.

Example usage:

```
{+hide-from-header{block}}
.example.com
```

Notes:

The keyword “block” will completely remove the header (not to be confused with the “+block” (configuration.html#BLOCK) action). Alternately, you can specify any value you prefer to send to the web server.

+hide-referer

Type:

Parameterized.

Typical uses:

Don’t send the “Referer:” (sic) HTTP header to the web site. Or, alternately send a forged header instead.

Possible values:

Prevent the header from being sent with the keyword, “block”. Or, “forge” a URL to one from the same server as the request. Or, set to user defined value of your choice.

Example usage:

```
{+hide-referer{forge}}
.example.com
```

Notes:

"forge" is the preferred option here, since some servers will not send images back otherwise.

"+hide-referrer" is an alternate spelling of "+hide-referer". It has the exact same parameters, and can be freely mixed with, "+hide-referer". ("referrer" is the correct English spelling, however the HTTP specification has a bug - it requires it to be spelled as "referer".)

+hide-user-agent

Type:

Parameterized.

Typical uses:

To change the "User-Agent:" header so web servers can't tell your browser type. Who's business is it anyway?

Possible values:

Any user defined string.

Example usage:

```
{+hide-user-agent{Netscape 6.1 (X11; I; Linux 2.4.18 i686)}}
.msn.com
```

Notes:

Warning! This breaks many web sites that depend on this in order to determine how the target browser will respond to various requests. Use with caution.

+handle-as-image

Type:

Boolean.

Typical uses:

To define what Privoxy should treat automatically as an image, and is an important ingredient of how ads are handled.

Possible values:

N/A

Example usage:

```
    {+handle-as-image}
/.*\.(gif|jpg|jpeg|png|bmp|ico)
```

Notes:

This only has meaning if the URL (or pattern) also is "+block"ed, in which case a user definable image can be sent rather than a HTML page. This is integral to the whole concept of ad blocking: the URL must match *both* a "+block" (configuration.html#BLOCK) rule, *and* "+handle-as-image". (See "+set-image-blocker" (configuration.html#SET-IMAGE-BLOCKER) below for control over what will actually be displayed by the browser.)

There is little reason to change the default definition for this action.

+set-image-blocker

Type:

Parameterized.

Typical uses:

Decide what to do with URLs that end up tagged with *both* "+block" (configuration.html#BLOCK) and "+handle-as-image" (configuration.html#HANDLE-AS-IMAGE), e.g an advertisement.

Possible values:

There are four available options: "-set-image-blocker" will send a HTML "blocked" page, usually resulting in a "broken image" icon. "+set-image-blocker{*blank*}" will send a 1x1 transparent GIF image. "+set-image-blocker{*pattern*}" will send a checkerboard type pattern (the default). And finally, "+set-image-blocker{*http://xyz.com*}" will send a HTTP temporary redirect to the specified image. This has the advantage of the icon being being cached by the browser, which will speed up the display.

Example usage:

```
    {+set-image-blocker{blank}}
.example.com
```

Notes:

If you want *invisible* ads, they need to meet criteria as matching both *images* and *blocked* actions. And then, “image-blocker” should be set to “blank” for invisibility. Note you cannot treat HTML pages as images in most cases. For instance, frames require an HTML page to display. So a frame that is an ad, typically cannot be treated as an image. Forcing an “image” in this situation just will not work reliably.

+limit-connect

Type:

Parameterized.

Typical uses:

By default, Privoxy only allows HTTP CONNECT requests to port 443 (the standard, secure HTTPS port). Use “+limit-connect” to disable this altogether, or to allow more ports.

Possible values:

Any valid port number, or port number range.

Example usages:

```
    +limit-connect{443}              # This is the default and need not be specified.
+limit-connect{80,443}           # Ports 80 and 443 are OK.
+limit-connect{-3, 7, 20-100, 500-}  # Port less than 3, 7, 20 to 100 and above 500 are OK.
```

Notes:

The CONNECT methods exists in HTTP to allow access to secure websites (https:// URLs) through proxies. It works very simply: the proxy connects to the server on the specified port, and then short-circuits its connections to the client *and* to the remote proxy. This can be a big security hole, since CONNECT-enabled proxies can be abused as TCP relays very easily.

If you want to allow CONNECT for more ports than this, or want to forbid CONNECT altogether, you can specify a comma separated list of ports and port ranges (the latter using dashes, with the minimum defaulting to 0 and max to 65K).

If you don't know what any of this means, there probably is no reason to change this one.

+prevent-compression

Type:

Boolean.

Typical uses:

Prevent the specified websites from compressing HTTP data.

Possible values:

N/A

Example usage:

```
    {+prevent-compression}
.example.com
```

Notes:

Some websites do this, which can be a problem for Privoxy, since "+filter" (configuration.html#FILTER), "+kill-popups" (configuration.html#KILL-POPUPS) and "+gif-deanimate" (configuration.html#GIF-DEANIMATE) will not work on compressed data. This will slow down connections to those websites, though. Default typically is to turn "prevent-compression" on.

+session-cookies-only

Type:

Boolean.

Typical uses:

Allow cookies for the current browser session *only*.

Possible values:

N/A

Example usage (disabling):

{-session-cookies-only}
.example.com

Notes:

If websites set cookies, “+session-cookies-only” will make sure they are erased when you exit and restart your web browser. This makes profiling cookies useless, but won’t break sites which require cookies so that you can log in for transactions. This is generally turned on for all sites, and is the recommended setting.

“+prevent-*-cookies” actions should be turned off as well (see below), for “+session-cookies-only” to work. Or, else no cookies will get through at all. For, “persistent” cookies that survive across browser sessions, see below as well.

+prevent-reading-cookies

Type:

Boolean.

Typical uses:

Explicitly prevent the web server from reading any cookies on your system.

Possible values:

N/A

Example usage:

{+prevent-reading-cookies}
.example.com

Notes:

Often used in conjunction with “+prevent-setting-cookies” to disable cookies completely. Note that “+session-cookies-only” (configuration.html#SESSION-COOKIES-ONLY) requires these to both be disabled (or else it never gets any cookies to cache).

For “persistent” cookies to work (i.e. they survive across browser sessions and reboots), all three cookie settings should be “off” for the specified sites.

+prevent-setting-cookies

Type:

Boolean.

Typical uses:

Explicitly block the web server from storing cookies on your system.

Possible values:

N/A

Example usage:

```
    {+prevent-setting-cookies}
.example.com
```

Notes:

Often used in conjunction with “+prevent-reading-cookies” to disable cookies completely (see above).

+kill-popups

Type:

Boolean.

Typical uses:

Stop those annoying JavaScript pop-up windows!

Possible values:

N/A

Example usage:

{+kill-popups}
.example.com

Notes:

"+kill-popups" uses a built in filter to disable pop-ups that use the `window.open()` function, etc. This is one of the first actions processed by Privoxy as it contacts the remote web server. This action is not always 100% reliable, and is supplemented by "+filter{*popups*}".

+send-vanilla-wafer

Type:

Boolean.

Typical uses:

Sends a cookie for every site stating that you do not accept any copyright on cookies sent to you, and asking them not to track you.

Possible values:

N/A

Example usage:

{+send-vanilla-wafer}
.example.com

Notes:

This action only applies if you are using a `jarfile` for saving cookies. Of course, this is a (relatively) unique header and could conceivably be used to track you.

+send-wafer

Type:

Multi-value.

Typical uses:

This allows you to send an arbitrary, user definable cookie.

Possible values:

User specified cookie name and corresponding value.

Example usage:

{+send-wafer{name=value}}
.example.com

Notes:

This can be specified multiple times in order to add as many cookies as you like.

Actions Examples

Note that the meaning of any of the above examples is reversed by preceding the action with a "-", in place of the "+". Also, that some actions are turned on in the default section of the actions file, and require little to no additional configuration. These are just "on". But, other actions that are turned on the default section *do typically require* exceptions to be listed in the lower sections of actions file. E.g. by default no URLs are "blocked" (i.e. in the default definitions of `default.action`). We need exceptions to this in order to enable ad blocking.

Some examples:

Turn off cookies by default, then allow a few through for specified sites (showing an excerpt from the "default" section of an actions file ONLY):

```
# Excerpt only:
# Allow cookies to and from the server, but
# for this browser session ONLY
{
  # other actions normally listed here...
  -prevent-setting-cookies \
  -prevent-reading-cookies \
```

```
  +session-cookies-only   \
}
/ # match all URLs

# Exceptions to the above, sites that benefit from persistent cookies
# that are saved from one browser session to the next.
{ -session-cookies-only }
 .javasoft.com
 .sun.com
 .yahoo.com
 .msdn.microsoft.com
 .redhat.com
```

Now turn off “fast redirects”, and then we allow two exceptions:

```
# Turn them off (excerpt only)!
{
 # other actions normally listed here...
 +fast-redirects
}
/ # match all URLs

# Reverse it for these two sites, which don’t work right without it.
{-fast-redirects}
 www.ukc.ac.uk/cgi-bin/wac\.cgi\?
 login.yahoo.com
```

Turn on page filtering according to rules in the defined sections of `default.filter`, and make one exception for Sourceforge:

```
# Run everything through the filter file, using only certain
# specified sections:
{
 # other actions normally listed here...
 +filter{html-annoyances} +filter{js-annoyances} +filter{kill-popups}\
 +filter{webbugs} +filter{nimda} +filter{banners-by-size}
}
```

```
/ #match all URLs

# Then disable filtering of code from all sourceforge domains!
{-filter}
 .sourceforge.net
```

Now some URLs that we want “blocked” (normally generates the “blocked” banner). Typically, the “block” action is off by default in the upper section of an actions file, then enabled against certain URLs and patterns in the lower part of the file. Many of these use regular expressions that will expand to match multiple URLs:

```
# Blocklist:
{+block}
ad*.
.*ads.
banner?.
count*.
/.*count(er)?\.(pl|cgi|exe|dll|asp|php[34]?)
/(?:.*/)?(publicite|werbung|rekla(ma|me|am)|annonse|maino(kset|nta|s)?)/
.hitbox.com
/.*/(ng)?adclient\.cgi
/.*/(plain|live|rotate)[-_.]?ads?/
/.*/abanners/
/autoads/
```

Note that many of these actions have the potential to cause a page to misbehave, possibly even not to display at all. There are many ways a site designer may choose to design his site, and what HTTP header content, and other criteria, he may depend on. There is no way to have hard and fast rules for all sites. See the Appendix for a brief example on troubleshooting actions.

Aliases

Custom “actions”, known to Privoxy as “aliases”, can be defined by combining other “actions”. These can in turn be invoked just like the built-in “actions”. Currently, an alias can contain any character except space, tab, “=”, “{” or “}”. But please use only “a”- “z”, “0”-“9”, “+”, and “-”. Alias names are not case sensitive, and *must be defined before other actions* in the actions file! And there can only be one set of

“aliases” defined per file. Each actions file may have its own aliases, but they are only visible within that file.

Now let’s define a few aliases:

```
 # Useful custom aliases we can use later. These must come first!
 {{alias}}
 +prevent-cookies = +prevent-setting-cookies +prevent-reading-cookies
 -prevent-cookies = -prevent-setting-cookies -prevent-reading-cookies
 fragile     = -block -prevent-cookies -filter -fast-redirects -hide-referer -
kill-popups
 shop        = -prevent-cookies -filter -fast-redirects
 +imageblock = +block +handle-as-image

 # Aliases defined from other aliases, for people who don’t like to type
 # too much:  ;-)
 c0 = +prevent-cookies
 c1 = -prevent-cookies
 #... etc.  Customize to your heart’s content.
```

Some examples using our “shop” and “fragile” aliases from above. These would appear in the lower sections of an actions file as exceptions to the default actions (as defined in the upper section):

```
 # These sites are very complex and require
 # minimal interference.
 {fragile}
  .office.microsoft.com
  .windowsupdate.microsoft.com
  .nytimes.com

 # Shopping sites - but we still want to block ads.
 {shop}
  .quietpc.com
  .worldpay.com   # for quietpc.com
  .scan.co.uk

 # These shops require pop-ups also
 {shop -kill-popups}
  .dabs.com
  .overclockers.co.uk
```

The “shop” and “fragile” aliases are often used for “problem” sites that require most actions to be disabled in order to function properly.

The Filter File

Any web page can be dynamically modified with the filter file. This modification can be removal, or re-writing, of any web page content, including tags and non-visible content. The default filter file is `default.filter`, located in the config directory.

This is potentially a very powerful feature, and requires knowledge of both “regular expression” and HTML in order create custom filters. But, there are a number of useful filters included with Privoxy for many common situations.

The included example file is divided into sections. Each section begins with the `FILTER` keyword, followed by the identifier for that section, e.g. “FILTER: webbugs”. Each section performs a similar type of filtering, such as “html-annoyances”.

This file uses regular expressions to alter or remove any string in the target page. The expressions can only operate on one line at a time. Some examples from the included default `default.filter`:

Stop web pages from displaying annoying messages in the status bar by deleting such references:

```
FILTER: html-annoyances

# New browser windows should be resizeable and have a location and status
# bar. Make it so.
#
s/resizable="?(no|0)"?/resizable=1/ig s/noresize/yesresize/ig
s/location="?(no|0)"?/location=1/ig s/status="?(no|0)"?/status=1/ig
s/scrolling="?(no|0|Auto)"?/scrolling=1/ig
s/menubar="?(no|0)"?/menubar=1/ig

# The <BLINK> tag was a crime!
#
s*<blink>|</blink>**ig

# Is this evil?
#
#s/framespacing="?(no|0)"?//ig
```

```
 #s/margin(height|width)=[0-9]*//gi
```

Just for kicks, replace any occurrence of “Microsoft” with “MicroSuck”, and have a little fun with topical buzzwords:

```
 FILTER: fun

 s/microsoft(?!.com)/MicroSuck/ig

 # Buzzword Bingo:
 #
 s/industry-leading|cutting-edge|award-winning/<font color=red><b>BINGO!</b></font>/ig
```

Kill those pesky little web-bugs:

```
 # webbugs: Squish WebBugs (1x1 invisible GIFs used for user tracking)
 FILTER: webbugs

 s/<img\s+[^>]*?(width|height)\s*=\s*['"]?1\D[^>]*?(width|height)\s*=\s*['"]?1(\D[^>]*?)?>
- Squished WebBug -->/sig
```

Templates

When Privoxy displays one of its internal pages, such as a 404 Not Found error page, it uses the appropriate template. On Linux, BSD, and Unix, these are located in `/etc/privoxy/templates` by default. These may be customized, if desired. `cgi-style.css` is used to control the HTML attributes (fonts, etc).

The default “Blocked” banner page with the bright red top banner, is called just “`blocked`”. This may be customized or replaced with something else if desired.

Contacting the Developers, Bug Reporting and Feature Requests

We value your feedback. However, to provide you with the best support, please note the following sections.

Get Support

To get support, use the Sourceforge Support Forum:

http://sourceforge.net/tracker/?group_id=11118&atid=211118

Report bugs

To submit bugs, use the Sourceforge Bug Forum:

http://sourceforge.net/tracker/?group_id=11118&atid=111118.

Make sure that the bug has not already been submitted. Please try to verify that it is a Privoxy bug, and not a browser or site bug first. If you are using your own custom configuration, please try the stock configs to see if the problem is a configuration related bug. And if not using the latest development snapshot, please try the latest one. Or even better, CVS sources. Please be sure to include the Privoxy version, platform, browser, any pertinent log data, any other relevant details (please be specific) and, if possible, some way to reproduce the bug.

Request new features

To submit ideas on new features, use the Sourceforge feature request forum:

http://sourceforge.net/tracker/?atid=361118&group_id=11118&func=browse.

Report ads or other filter problems

You can also send feedback on websites that Privoxy has problems with. Please bookmark the following link: “Privoxy - Submit Filter Feedback” (javascript:w=Math.floor(screen.width/2);h=Math.floor(screen.height*0.9);void(window.open(’http://www.privoxy.org/acti
Once you surf to a page with problems, use the bookmark to send us feedback. We will look into the issue as soon as possible.

New, improved `default.action` files will occasionally be made available based on your feedback. These will be announced on the ijbswa-announce (http://lists.sourceforge.net/lists/listinfo/ijbswa-announce) list.

Other

For any other issues, feel free to use the mailing lists:

http://sourceforge.net/mail/?group_id=11118.

Anyone interested in actively participating in development and related discussions can also join the appropriate mailing list. Archives are available, too. See the page on Sourceforge.

Copyright and History

Copyright

Privoxy is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

History

Privoxy is evolved, and derived from, the Internet Junkbuster, with many improvments and enhancements over the original.

Junkbuster was originally written by Anonymous Coders and Junkbusters Corporation (http://www.junkbusters.com), and was released as free open-source software under the GNU GPL. Stefan Waldherr (http://www.waldherr.org/junkbuster/) made many improvements, and started the SourceForge project Privoxy (http://sourceforge.net/projects/ijbswa/) to rekindle development. There are now several active developers contributing. The last stable release of Junkbuster was v2.0.2, which has now grown whiskers ;-).

See Also

Other references and sites of interest to Privoxy users:

http://www.privoxy.org/, The Privoxy Home page.
http://sourceforge.net/projects/ijbswa, the Project Page for Privoxy on Sourceforge (http://sourceforge.net).
http://p.p/, access Privoxy from your browser. Alternately, http://config.privoxy.org may work in some situations where the
http://p.p/, and select “actions file feedback system” (javascript:w=Math.floor(screen.width/2);h=Math.floor(screen.height
http://www.junkbusters.com/ht/en/cookies.html
http://www.waldherr.org/junkbuster/
http://privacy.net/analyze/
http://www.squid-cache.org/

Appendix

Regular Expressions

Privoxy can use “regular expressions” in various config files. Assuming support for “pcre” (Perl Compatible Regular Expressions) is compiled in, which is the default. Such configuration directives do not require regular expressions, but they can be used to increase flexibility by matching a pattern with wild-cards against URLs.

If you are reading this, you probably don’t understand what “regular expressions” are, or what they can do. So this will be a very brief introduction only. A full explanation would require a book ;-)

“Regular expressions” is a way of matching one character expression against another to see if it matches or not. One of the “expressions” is a literal string of readable characters (letter, numbers, etc), and the other is a complex string of literal characters combined with wild-cards, and other special characters, called meta-characters. The “meta-characters” have special meanings and are used to build the complex pattern to be matched against. Perl Compatible Regular Expressions is an enhanced form of the regular expression language with backward compatibility.

To make a simple analogy, we do something similar when we use wild-card characters when listing files with the **dir** command in DOS. `*.*` matches all filenames. The “special” character here is the asterisk which matches any and all characters. We can be more specific and use `?` to match just individual characters. So “dir file?.text” would match “file1.txt”, “file2.txt”, etc. We are pattern matching, using a similar technique to “regular expressions”!

Regular expressions do essentially the same thing, but are much, much more powerful. There are many more “special characters” and ways of building complex patterns however. Let’s look at a few of the common ones, and then some examples:

. - Matches any single character, e.g. “a”, “A”, “4”, “:”, or “@”.

? - The preceding character or expression is matched ZERO or ONE times. Either/or.

+ - The preceding character or expression is matched ONE or MORE times.

* - The preceding character or expression is matched ZERO or MORE times.

\ - The “escape” character denotes that the following character should be taken literally. This is used where one of the spec

[] - Characters enclosed in brackets will be matched if any of the enclosed characters are encountered. For instance, “[0-9]

() - parentheses are used to group a sub-expression, or multiple sub-expressions.

| - The “bar” character works like an “or” conditional statement. A match is successful if the sub-expression on either side

s/string1/string2/g - This is used to rewrite strings of text. “string1” is replaced by “string2” in this example. There must o

These are just some of the ones you are likely to use when matching URLs with Privoxy, and is a long way from a definitive list. This is enough to get us started with a few simple examples which may be more illuminating:

/./banners/.** - A simple example that uses the common combination of “.” and “*” to denote any character, zero or more times. In other words, any string at all. So we start with a literal forward slash, then our regular expression pattern (“.*”) another literal forward slash, the string “banners”, another forward slash, and lastly another “.*”. We are building a directory path here. This will match any file with the path that has a directory named “banners” in it. The “.*” matches any characters, and this could conceivably be more forward slashes, so it might expand into a much longer looking path. For example, this could match: “/eye/hate/spammers/banners/annoy_me_please.gif”, or just “/banners/annoying.html”, or almost an infinite number of other possible combinations, just so it has “banners” in the path somewhere.

A now something a little more complex:

/./adv((er)?ts?|ertis(ing|ements?))?/* - We have several literal forward slashes again (“/”), so we are building another expression that is a file path statement. We have another “.*”, so we are matching against any conceivable sub-path, just so it matches our expression. The only true literal that *must match* our pattern is adv, together with the forward slashes. What comes after the “adv” string is the interesting part.

Remember the “?” means the preceding expression (either a literal character or anything grouped with “(...)” in this case) can exist or not, since this means either zero or one match. So “((er)?ts?|ertis(ing|ements?))” is optional, as are the individual sub-expressions: “(er)”, “(ing|ements?)”, and the “s”. The “|” means “or”. We have two of those. For instance, “(ing|ements?)”, can expand to match either “ing” *OR* “ements?”. What is being done here, is an attempt at matching as many variations of “advertisement”, and similar, as possible. So this would expand to match just “adv”, or “advert”, or “adverts”, or “advertising”, or “advertisement”, or “advertisements”. You get the idea. But it would not match “advertizements” (with a “z”). We could fix that by changing our regular expression to: “/.*/adv((er)?ts?|erti(s|z)(ing|ements?))?/”, which would then match either spelling.

/./advert[0-9]+\.(gif|jpe?g)* - Again another path statement with forward slashes. Anything in the square brackets “[]” can be matched. This is using “0-9” as a shorthand expression to mean any digit one through nine. It is the same as saying “0123456789”. So any digit matches. The “+” means one or more of the preceding expression must be included. The preceding expression here is what is in the square brackets -- in this case, any digit one through nine. Then, at the end, we have a grouping: “(gif|jpe?g)”. This includes a “|”, so this needs to match the expression on either side of that bar character also. A simple “gif” on one side, and the other side will in turn match either “jpeg” or “jpg”, since the “?” means the letter “e” is optional and can be matched once or not at all. So we are building an expression here to match image GIF or JPEG type image file. It must include the literal string “advert”, then one or more digits, and a “.” (which is now a literal, and not a special character, since it is escaped

with “\”), and lastly either “gif”, or “jpeg”, or “jpg”. Some possible matches would include: “//advert1.jpg”, “/nasty/ads/advert1234.gif”, “/banners/from/hell/advert99.jpg”. It would not match “advert1.gif” (no leading slash), or “/adverts232.jpg” (the expression does not include an “s”), or “/advert1.jsp” (“jsp” is not in the expression anywhere).

`s/microsoft(?!.com)/MicroSuck/i` - This is a substitution. “MicroSuck” will replace any occurrence of “microsoft”. The “i” at the end of the expression means ignore case. The “(?!.com)” means the match should fail if “microsoft” is followed by “.com”. In other words, this acts like a “NOT” modifier. In case this is a hyperlink, we don’t want to break it ;-).

We are barely scratching the surface of regular expressions here so that you can understand the default Privoxy configuration files, and maybe use this knowledge to customize your own installation. There is much, much more that can be done with regular expressions. Now that you know enough to get started, you can learn more on your own :/

More reading on Perl Compatible Regular expressions: http://www.perldoc.com/perl5.6/pod/perlre.html

Privoxy’s Internal Pages

Since Privoxy proxies each requested web page, it is easy for Privoxy to trap certain special URLs. In this way, we can talk directly to Privoxy, and see how it is configured, see how our rules are being applied, change these rules and other configuration options, and even turn Privoxy’s filtering off, all with a web browser.

The URLs listed below are the special ones that allow direct access to Privoxy. Of course, Privoxy must be running to access these. If not, you will get a friendly error message. Internet access is not necessary either.

- Privoxy main page:

  http://config.privoxy.org/

  Alternately, this may be reached at http://p.p/, but this variation may not work as reliably as the above in some configurations.

- Show information about the current configuration, including viewing and editing of actions files:

  http://config.privoxy.org/show-status

- Show the source code version numbers:

http://config.privoxy.org/show-version

- Show the browser's request headers:

  http://config.privoxy.org/show-request

- Show which actions apply to a URL and why:

  http://config.privoxy.org/show-url-info

- Toggle Privoxy on or off. In this case, "Privoxy" continues to run, but only as a pass-through proxy, with no actions taking place:

  http://config.privoxy.org/toggle

  Short cuts. Turn off, then on:

  http://config.privoxy.org/toggle?set=disable

  http://config.privoxy.org/toggle?set=enable

These may be bookmarked for quick reference. See next.

Bookmarklets

Below are some "bookmarklets" to allow you to easily access a "mini" version of some of Privoxy's special pages. They are designed for MS Internet Explorer, but should work equally well in Netscape, Mozilla, and other browsers which support JavaScript. They are designed to run directly from your bookmarks - not by clicking the links below (although that should work for testing).

To save them, right-click the link and choose "Add to Favorites" (IE) or "Add Bookmark" (Netscape). You will get a warning that the bookmark "may not be safe" - just click OK. Then you can run the Bookmarklet directly from your favorites/bookmarks. For even faster access, you can put them on the "Links" bar (IE) or the "Personal Toolbar" (Netscape), and run them with a single click.

- Privoxy - Enable
  (javascript:void(window.open('http://config.privoxy.org/toggle?mini=y&set=enabled','ijbstatus','width=250,height=100,
- Privoxy - Disable
  (javascript:void(window.open('http://config.privoxy.org/toggle?mini=y&set=disabled','ijbstatus','width=250,height=100
- Privoxy - Toggle Privoxy
  (javascript:void(window.open('http://config.privoxy.org/toggle?mini=y&set=toggle','ijbstatus','width=250,height=100,re
  (Toggles between enabled and disabled)
- Privoxy- View Status
  (javascript:void(window.open('http://config.privoxy.org/toggle?mini=y','ijbstatus','width=250,height=2,resizable=yes,sc
- Privoxy - Submit Filter Feedback
  (javascript:w=Math.floor(screen.width/2);h=Math.floor(screen.height*0.9);void(window.open('http://www.privoxy.org/a

Credit: The site which gave me the general idea for these bookmarklets is www.bookmarklets.com (http://www.bookmarklets.com). They have more information about bookmarklets.

Chain of Events

Let's take a quick look at the basic sequence of events when a web page is requested by your browser and Privoxy is on duty:

- First, your web browser requests a web page. The browser knows to send the request to Privoxy, which will in turn, relay the request to the remote web server after passing the following tests:
- Privoxy traps any request for its own internal CGI pages (e.g http://p.p/) and sends the CGI page back to the browser.
- Next, Privoxy checks to see if the URL matches any "+block" (configuration.html#BLOCK) patterns. If so, the URL is then blocked, and the remote web server will not be contacted. "+handle-as-image" (configuration.html#HANDLE-AS-IMAGE) is then checked and if it does not match, an HTML "BLOCKED" page is sent back. Otherwise, if it does match, an image is returned. The type of image depends on the setting of "+set-image-blocker" (configuration.html#SET-IMAGE-BLOCKER) (blank, checkerboard pattern, or an HTTP redirect to an image elsewhere).
- Untrusted URLs are blocked. If URLs are being added to the `trust` file, then that is done.
- If the URL pattern matches the "+fast-redirects" (configuration.html#FAST-REDIRECTS) action, it is then processed. Unwanted parts of the requested URL are stripped.

- Now the rest of the client browser's request headers are processed. If any of these match any of the relevant actions (e.g. "+hide-user-agent" (configuration.html#HIDE-USER-AGENT), etc.), headers are suppressed or forged as determined by these actions and their parameters.
- Now the web server starts sending its response back (i.e. typically a web page and related data).
- First, the server headers are read and processed to determine, among other things, the MIME type (document type) and encoding. The headers are then filtered as deterimed by the "+prevent-setting-cookies" (configuration.html#PREVENT-SETTING-COOKIES), "+session-cookies-only" (configuration.html#SESSION-COOKIES-ONLY), and "+downgrade-http-version" (configuration.html#DOWNGRADE-HTTP-VERSION) actions.
- If the "+kill-popups" (configuration.html#KILL-POPUPS) action applies, and it is an HTML or JavaScript document, the popup-code in the response is filtered on-the-fly as it is received.
- If a "+filter" (configuration.html#FILTER) or "+deanimate-gifs" (configuration.html#DEANIMATE-GIFS) action applies (and the document type fits the action), the rest of the page is read into memory (up to a configurable limit). Then the filter rules (from `default.filter`) are processed against the buffered content. Filters are applied in the order they are specified in the `default.filter` file. Animated GIFs, if present, are reduced to either the first or last frame, depending on the action setting.The entire page, which is now filtered, is then sent by Privoxy back to your browser.

  If neither "+filter" (configuration.html#FILTER) or "+deanimate-gifs" (configuration.html#DEANIMATE-GIFS) matches, then Privoxy passes the raw data through to the client browser as it becomes available.

- As the browser receives the now (probably filtered) page content, it reads and then requests any URLs that may be embedded within the page source, e.g. ad images, stylesheets, JavaScript, other HTML documents (e.g. frames), sounds, etc. For each of these objects, the browser issues a new request. And each such request is in turn processed as above. Note that a complex web page may have many such embedded URLs.

Anatomy of an Action

The way Privoxy applies "actions" (configuration.html#ACTIONS) and "filters" (configuration.html#FILTER) to any given URL can be complex, and not always so easy to understand what is happening. And sometimes we need to be able to *see* just what Privoxy is doing. Especially, if something Privoxy is doing is causing us a problem inadvertently. It can be a little daunting to look at the

actions and filters files themselves, since they tend to be filled with “regular expressions” whose consequences are not always so obvious.

One quick test to see if Privoxy is causing a problem or not, is to disable it temporarily. This should be the first troubleshooting step. See the Bookmarklets section on a quick and easy way to do this (be sure to flush caches afterward!).

Privoxy also provides the http://config.privoxy.org/show-url-info page that can show us very specifically how actions are being applied to any given URL. This is a big help for troubleshooting.

First, enter one URL (or partial URL) at the prompt, and then Privoxy will tell us how the current configuration will handle it. This will not help with filtering effects (i.e. the “+filter” (configuration.html#FILTER) action) from the `default.filter` file since this is handled very differently and not so easy to trap! It also will not tell you about any other URLs that may be embedded within the URL you are testing. For instance, images such as ads are expressed as URLs within the raw page source of HTML pages. So you will only get info for the actual URL that is pasted into the prompt area -- not any sub-URLs. If you want to know about embedded URLs like ads, you will have to dig those out of the HTML source. Use your browser’s “View Page Source” option for this. Or right click on the ad, and grab the URL.

Let’s try an example, google.com (http://google.com), and look at it one section at a time:

```
 Matches for http://google.com:

--- File standard ---
(no matches in this file)

--- File default ---

{ -add-header -block +deanimate-gifs{last} -downgrade-http-version +fast-redirects
 -filter{popups} -filter{fun} -filter{shockwave-flash} -filter{crude-parental}
 +filter{html-annoyances} +filter{js-annoyances} +filter{content-cookies}
 +filter{webbugs} +filter{refresh-tags} +filter{nimda} +filter{banners-by-size}
 +hide-forwarded-for-headers +hide-from-header{block} +hide-referer{forge}
 -hide-user-agent -handle-as-image +set-image-blocker{pattern} -limit-connect
 +prevent-compression +session-cookies-only -prevent-reading-cookies
 -prevent-setting-cookies -kill-popups -send-vanilla-wafer -send-wafer }
/

 { -session-cookies-only }
 .google.com

 { -fast-redirects }
 .google.com
```

```
--- File user ---
(no matches in this file)
```

This tells us how we have defined our “actions” (configuration.html#ACTIONS), and which ones match for our example, “google.com”. The first listing is any matches for the `standard.action` file. No hits at all here on “standard”. Then next is “default”, or our `default.action` file. The large, multi-line listing, is how the actions are set to match for all URLs, i.e. our default settings. If you look at your “actions” file, this would be the section just below the “aliases” section near the top. This will apply to all URLs as signified by the single forward slash at the end of the listing -- “/”.

But we can define additional actions that would be exceptions to these general rules, and then list specific URLs (or patterns) that these exceptions would apply to. Last match wins. Just below this then are two explicit matches for “.google.com”. The first is negating our previous cookie setting, which was for “+session-cookies-only” (configuration.html#SESSION-COOKIES-ONLY) (i.e. not persistent). So we will allow persistent cookies for google. The second turns *off* any “+fast-redirects” (configuration.html#FAST-REDIRECTS) action, allowing this to take place unmolested. Note that there is a leading dot here -- “.google.com”. This will match any hosts and sub-domains, in the google.com domain also, such as “www.google.com”. So, apparently, we have these two actions defined somewhere in the lower part of our `default.action` file, and “google.com” is referenced somewhere in these latter sections.

Then, for our `user.action` file, we again have no hits.

And finally we pull it all together in the bottom section and summarize how Privoxy is applying all its “actions” to “google.com”:

```
Final results:
-add-header -block +deanimate-gifs{last} -downgrade-http-version -fast-redirects
-filter{popups} -filter{fun} -filter{shockwave-flash} -filter{crude-parental}
+filter{html-annoyances} +filter{js-annoyances} +filter{content-cookies}
+filter{webbugs} +filter{refresh-tags} +filter{nimda} +filter{banners-by-size}
+hide-forwarded-for-headers +hide-from-header{block} +hide-referer{forge}
-hide-user-agent -handle-as-image +set-image-blocker{pattern} -limit-connect
+prevent-compression -session-cookies-only -prevent-reading-cookies
-prevent-setting-cookies -kill-popups -send-vanilla-wafer -send-wafer
```

Notice the only difference here to the previous listing, is to “fast-redirects” and “session-cookies-only”.

Now another example, “ad.doubleclick.net”:

```
{ +block +handle-as-image }
```

```
  .ad.doubleclick.net

 { +block +handle-as-image }
  ad*.

 { +block +handle-as-image }
  .doubleclick.net
```

We'll just show the interesting part here, the explicit matches. It is matched three different times. Each as an "+block +handle-as-image", which is the expanded form of one of our aliases that had been defined as: "+imageblock". ("Aliases" (configuration.html#ALIASES) are defined in the first section of the actions file and typically used to combine more than one action.)

Any one of these would have done the trick and blocked this as an unwanted image. This is unnecessarily redundant since the last case effectively would also cover the first. No point in taking chances with these guys though ;-) Note that if you want an ad or obnoxious URL to be invisible, it should be defined as "ad.doubleclick.net" is done here -- as both a "+block" (configuration.html#BLOCK) *and* an "+handle-as-image" (configuration.html#HANDLE-AS-IMAGE). The custom alias "+imageblock" just simplifies the process and make it more readable.

One last example. Let's try "http://www.rhapsodyk.net/adsl/HOWTO/". This one is giving us problems. We are getting a blank page. Hmmm...

```
 Matches for http://www.rhapsodyk.net/adsl/HOWTO/:

 { -add-header -block +deanimate-gifs -downgrade-http-version +fast-redirects
   +filter{html-annoyances} +filter{js-annoyances} +filter{kill-popups}
   +filter{webbugs} +filter{nimda} +filter{banners-by-size} +filter{hal}
   +filter{fun} +hide-forwarded-for-headers +hide-from-header{block}
   +hide-referer{forge} -hide-user-agent -handle-as-image +set-image-blocker{blank}
   +prevent-compression +session-cookies-only -prevent-setting-cookies
   -prevent-reading-cookies +kill-popups -send-vanilla-wafer -send-wafer }
   /

 { +block +handle-as-image }
  /ads
```

Ooops, the "/adsl/" is matching "/ads"! But we did not want this at all! Now we see why we get the blank page. We could now add a new action below this that explicitly does *not* block ("{-block}") paths with "adsl". There are various ways to handle such exceptions. Example:

```
{ -block }
 /adsl
```

Now the page displays ;-) Be sure to flush your browser's caches when making such changes. Or, try using `Shift+Reload`.

But now what about a situation where we get no explicit matches like we did with:

```
{ +block +handle-as-image }
/ads
```

That actually was very telling and pointed us quickly to where the problem was. If you don't get this kind of match, then it means one of the default rules in the first section is causing the problem. This would require some guesswork, and maybe a little trial and error to isolate the offending rule. One likely cause would be one of the "{+filter}" actions. Try adding the URL for the site to one of aliases that turn off "+filter":

```
{shop}
.quietpc.com
.worldpay.com   # for quietpc.com
.jungle.com
.scan.co.uk
.forbes.com
```

"{shop}" is an "alias" that expands to "{ -filter -session-cookies-only }". Or you could do your own exception to negate filtering:

```
{-filter}
.forbes.com
```

This would probably be most appropriately put in `user.action`, for local site exceptions.

"{fragile}" is an alias that disables most actions. This can be used as a last resort for problem sites. Remember to flush caches! If this still does not work, you will have to go through the remaining actions one by one to find which one(s) is causing the problem.