NAME

tor − The second−generation onion router

SYNOPSIS

tor [*OPTION value*]...

DESCRIPTION

Tor is a connection−oriented anonymizing communication service. Users choose a source−routed path through a set of nodes, and negotiate a "virtual circuit" through the network. Each node in a virtual circuit knows its predecessor and successor nodes, but no other nodes. Traffic flowing down the circuit is unwrapped by a symmetric key at each node, which reveals the downstream node.

Basically, Tor provides a distributed network of servers or relays ("onion routers"). Users bounce their TCP streams, including web traffic, ftp, ssh, etc., around the network, so that recipients, observers, and even the relays themselves have difficulty tracking the source of the stream.

Note

By default, **tor** acts as a client only. To help the network by providing bandwidth as a relay, change the **ORPort** configuration option as mentioned below. Please also consult the documentation on the Tor Project's website.

COMMAND−LINE OPTIONS

Tor has a powerful command−line interface. This section lists optional arguments you can specify at the command line using the **tor** command.

Configuration options can be specified on the command line in the format **−−***OptionName OptionValue*, on the command line in the format *OptionName OptionValue*, or in a configuration file. For instance, you can tell Tor to start listening for SOCKS connections on port 9999 by passing either **−−SocksPort 9999** or **SocksPort 9999** on the command line, or by specifying **SocksPort 9999** in the configuration file. On the command line, quote option values that contain spaces. For instance, if you want Tor to log all debugging messages to **debug.log**, you must specify **−−Log "debug file debug.log"**.

Note

Configuration options on the command line override those in configuration files. See **THE CONFIGURATION FILE FORMAT** for more information.

The following options in this section are only recognized on the **tor** command line, not in a configuration file.

−h, **−−help**
Display a short help message and exit.

−f, **−−torrc−file** *FILE*
Specify a new configuration file to contain further Tor configuration options, or pass **−** to make Tor read its configuration from standard input. (Default: **/etc/tor/torrc**, or **$HOME/.torrc** if that file is not found)

−−allow−missing−torrc
Allow the configuration file specified by **−f** to be missing, if the defaults−torrc file (see below) is accessible.

−−defaults−torrc *FILE*
Specify a file in which to find default values for Tor options. The contents of this file are overridden by those in the regular configuration file, and by those on the command line. (Default: **/etc/tor/torrc−defaults**.)

−−ignore−missing−torrc
Specify that Tor should treat a missing torrc file as though it were empty. Ordinarily, Tor does this for missing default torrc files, but not for those specified on the command line.

−−hash−password *PASSWORD*
Generate a hashed password for control port access.

−−list−fingerprint [*key type*]
Generate your keys and output your nickname and fingerprint. Optionally, you can specify the key type as rsa (default) or ed25519.

−−verify−config
Verify whether the configuration file is valid.

−−dump−config short|**full**
Write a list of Tor's configured options to standard output. When the short flag is selected, only write the options that are different from their default values When full is selected, write every option.

−−service install [**−−options** *command−line options*]
Install an instance of Tor as a Windows service, with the provided command−line options. Current instructions can be found at https://www.torproject.org/docs/faq#NTService

−−service remove|**start**|**stop**
Remove, start, or stop a configured Tor Windows service.

−−nt−service
Used internally to implement a Windows service.

−−list−torrc−options
List all valid options.

−−list−deprecated−options
List all valid options that are scheduled to become obsolete in a future version. (This is a warning, not a promise.)

−−list−modules
List whether each optional module has been compiled into Tor. (Any module not listed is not optional in this version of Tor.)

−−version
Display Tor version and exit. The output is a single line of the format "Tor version [version number]." (The version number format is as specified in version−spec.txt.)

−−quiet|**−−hush**
Override the default console logging behavior. By default, Tor starts out logging messages at level "notice" and higher to the console. It stops doing so after it parses its configuration, if the configuration tells it to log anywhere else. These options override the default console logging behavior. Use the **−−hush** option if you want Tor to log only warnings and errors to the console, or use the **−−quiet** option if you want Tor not to log to the console at all.

−−keygen [**−−newpass**]
Running **tor −−keygen** creates a new ed25519 master identity key for a relay, or only a fresh temporary signing key and certificate, if you already have a master key. Optionally, you can encrypt the master identity key with a passphrase. When Tor asks you for a passphrase and you don't want to encrypt the master key, just don't enter any passphrase when asked.

Use the **−−newpass** option with **−−keygen** only when you need to add, change, or remove a passphrase on an existing ed25519 master identity key. You will be prompted for the old passphrase (if any), and the new passphrase (if any).

Note
When generating a master key, you may want to use **−−DataDirectory** to control where the keys and certificates will be stored, and **−−SigningKeyLifetime** to control their lifetimes. See SERVER OPTIONS to learn more about the behavior of these options. You must have write access to the specified DataDirectory.

To use the generated files, you must copy them to the *DataDirectory*/**keys** directory of your Tor daemon, and make sure that they are owned by the user actually running the Tor daemon on your system.

−−passphrase−fd *FILEDES*

File descriptor to read the passphrase from. Note that unlike with the tor−gencert program, the entire file contents are read and used as the passphrase, including any trailing newlines. If the file descriptor is not specified, the passphrase is read from the terminal by default.

−−key−expiration [*purpose*] [**−−format iso8601**|**timestamp**]
The *purpose* specifies which type of key certificate to determine the expiration of. The only currently recognised *purpose* is "sign".

Running **tor −−key−expiration sign** will attempt to find your signing key certificate and will output, both in the logs as well as to stdout. The optional **−−format** argument lets you specify the time format. Currently, **iso8601** and **timestamp** are supported. If **−−format** is not specified, the signing key certificate's expiration time will be in ISO−8601 format. For example, the output sent to stdout will be of the form: "signing−cert−expiry: 2017−07−25 08:30:15 UTC". If **−−format timestamp** is specified, the signing key certificate's expiration time will be in Unix timestamp format. For example, the output sent to stdout will be of the form: "signing−cert−expiry: 1500971415".

−−dbg−...
Tor may support other options beginning with the string "dbg". These are intended for use by developers to debug and test Tor. They are not supported or guaranteed to be stable, and you should probably not use them.

THE CONFIGURATION FILE FORMAT

All configuration options in a configuration are written on a single line by default. They take the form of an option name and a value, or an option name and a quoted value (option value or option "value"). Anything after a # character is treated as a comment. Options are case−insensitive. C−style escaped characters are allowed inside quoted values. To split one configuration entry into multiple lines, use a single backslash character (\) before the end of the line. Comments can be used in such multiline entries, but they must start at the beginning of a line.

Configuration options can be imported from files or folders using the %include option with the value being a path. This path can have wildcards. Wildcards are expanded first, then sorted using lexical order. Then, for each matching file or folder, the following rules are followed: if the path is a file, the options from the file will be parsed as if they were written where the %include option is. If the path is a folder, all files on that folder will be parsed following lexical order. Files starting with a dot are ignored. Files in subfolders are ignored. The %include option can be used recursively. New configuration files or directories cannot be added to already running Tor instance if **Sandbox** is enabled.

The supported wildcards are * meaning any number of characters including none and ? meaning exactly one character. These characters can be escaped by preceding them with a backslash, except on Windows. Files starting with a dot are not matched when expanding wildcards unless the starting dot is explicitly in the pattern, except on Windows.

By default, an option on the command line overrides an option found in the configuration file, and an option in a configuration file overrides one in the defaults file.

This rule is simple for options that take a single value, but it can become complicated for options that are allowed to occur more than once: if you specify four SocksPorts in your configuration file, and one more SocksPort on the command line, the option on the command line will replace *all* of the SocksPorts in the configuration file. If this isn't what you want, prefix the option name with a plus sign (+), and it will be appended to the previous set of options instead. For example, setting SocksPort 9100 will use only port 9100, but setting +SocksPort 9100 will use ports 9100 and 9050 (because this is the default).

Alternatively, you might want to remove every instance of an option in the configuration file, and not replace it at all: you might want to say on the command line that you want no SocksPorts at all. To do that, prefix the option name with a forward slash (/). You can use the plus sign (+) and the forward slash (/) in

the configuration file and on the command line.

GENERAL OPTIONS

AccelDir *DIR*

Specify this option if using dynamic hardware acceleration and the engine implementation library resides somewhere other than the OpenSSL default. Can not be changed while tor is running.

AccelName *NAME*

When using OpenSSL hardware crypto acceleration attempt to load the dynamic engine of this name. This must be used for any dynamic hardware engine. Names can be verified with the openssl engine command. Can not be changed while tor is running.

If the engine name is prefixed with a "!", then Tor will exit if the engine cannot be loaded.

AlternateBridgeAuthority [*nickname*] [**flags**] *ipv4address*:*port fingerprint*, **AlternateDirAuthority** [*nickname*] [**flags**] *ipv4address*:*port fingerprint*

These options behave as DirAuthority, but they replace fewer of the default directory authorities. Using AlternateDirAuthority replaces the default Tor directory authorities, but leaves the default bridge authorities in place. Similarly, AlternateBridgeAuthority replaces the default bridge authority, but leaves the directory authorities alone.

AvoidDiskWrites 0|**1**

If non−zero, try to write to disk less frequently than we would otherwise. This is useful when running on flash memory or other media that support only a limited number of writes. (Default: 0)

BandwidthBurst *N* **bytes**|**KBytes**|**MBytes**|**GBytes**|**TBytes**|**KBits**|**MBits**|**GBits**|**TBits**

Limit the maximum token bucket size (also known as the burst) to the given number of bytes in each direction. (Default: 1 GByte)

BandwidthRate *N* **bytes**|**KBytes**|**MBytes**|**GBytes**|**TBytes**|**KBits**|**MBits**|**GBits**|**TBits**

A token bucket limits the average incoming bandwidth usage on this node to the specified number of bytes per second, and the average outgoing bandwidth usage to that same value. If you want to run a relay in the public network, this needs to be *at the very least* 75 KBytes for a relay (that is, 600 kbits) or 50 KBytes for a bridge (400 kbits) — but of course, more is better; we recommend at least 250 KBytes (2 mbits) if possible. (Default: 1 GByte)

Note that this option, and other bandwidth−limiting options, apply to TCP data only: They do not count TCP headers or DNS traffic.

Tor uses powers of two, not powers of ten, so 1 GByte is 1024*1024*1024 bytes as opposed to 1 billion bytes.

With this option, and in other options that take arguments in bytes, KBytes, and so on, other formats are also supported. Notably, "KBytes" can also be written as "kilobytes" or "kb"; "MBytes" can be written as "megabytes" or "MB"; "kbits" can be written as "kilobits"; and so forth. Case doesn’t matter. Tor also accepts "byte" and "bit" in the singular. The prefixes "tera" and "T" are also recognized. If no units are given, we default to bytes. To avoid confusion, we recommend writing "bytes" or "bits" explicitly, since it’s easy to forget that "B" means bytes, not bits.

CacheDirectory *DIR*

Store cached directory data in DIR. Can not be changed while tor is running. (Default: uses the value of DataDirectory.)

CacheDirectoryGroupReadable 0|**1**|**auto**

If this option is set to 0, don’t allow the filesystem group to read the CacheDirectory. If the option is

set to 1, make the CacheDirectory readable by the default GID. If the option is "auto", then we use the setting for DataDirectoryGroupReadable when the CacheDirectory is the same as the DataDirectory, and 0 otherwise. (Default: auto)

CircuitPriorityHalflife *NUM*
If this value is set, we override the default algorithm for choosing which circuit's cell to deliver or relay next. It is delivered first to the circuit that has the lowest weighted cell count, where cells are weighted exponentially according to this value (in seconds). If the value is −1, it is taken from the consensus if possible else it will fallback to the default value of 30. Minimum: 1, Maximum: 2147483647. This can be defined as a float value. This is an advanced option; you generally shouldn't have to mess with it. (Default: −1)

ClientTransportPlugin *transport* socks4|socks5 *IP*:*PORT*, **ClientTransportPlugin** *transport* exec *path−to−binary* [options]
In its first form, when set along with a corresponding Bridge line, the Tor client forwards its traffic to a SOCKS−speaking proxy on "IP:PORT". (IPv4 addresses should written as−is; IPv6 addresses should be wrapped in square brackets.) It's the duty of that proxy to properly forward the traffic to the bridge.

In its second form, when set along with a corresponding Bridge line, the Tor client launches the pluggable transport proxy executable in *path−to−binary* using *options* as its command−line options, and forwards its traffic to it. It's the duty of that proxy to properly forward the traffic to the bridge. (Default: none)

ConnLimit *NUM*
The minimum number of file descriptors that must be available to the Tor process before it will start. Tor will ask the OS for as many file descriptors as the OS will allow (you can find this by "ulimit −H −n"). If this number is less than ConnLimit, then Tor will refuse to start.

Tor relays need thousands of sockets, to connect to every other relay. If you are running a private bridge, you can reduce the number of sockets that Tor uses. For example, to limit Tor to 500 sockets, run "ulimit −n 500" in a shell. Then start tor in the same shell, with **ConnLimit 500**. You may also need to set **DisableOOSCheck 0**.

Unless you have severely limited sockets, you probably don't need to adjust **ConnLimit** itself. It has no effect on Windows, since that platform lacks getrlimit(). (Default: 1000)

ConstrainedSockets 0|**1**
If set, Tor will tell the kernel to attempt to shrink the buffers for all sockets to the size specified in **ConstrainedSockSize**. This is useful for virtual servers and other environments where system level TCP buffers may be limited. If you're on a virtual server, and you encounter the "Error creating network socket: No buffer space available" message, you are likely experiencing this problem.

The preferred solution is to have the admin increase the buffer pool for the host itself via /proc/sys/net/ipv4/tcp_mem or equivalent facility; this configuration option is a second−resort.

The DirPort option should also not be used if TCP buffers are scarce. The cached directory requests consume additional sockets which exacerbates the problem.

You should **not** enable this feature unless you encounter the "no buffer space available" issue. Reducing the TCP buffers affects window size for the TCP stream and will reduce throughput in proportion to round trip time on long paths. (Default: 0)

ConstrainedSockSize *N* **bytes**|**KBytes**
When **ConstrainedSockets** is enabled the receive and transmit buffers for all sockets will be set to this limit. Must be a value between 2048 and 262144, in 1024 byte increments. Default of 8192 is recommended.

ControlPort [*address***:**]*port*|**unix:***path*|**auto** [*flags*]
If set, Tor will accept connections on this port and allow those connections to control the Tor process using the Tor Control Protocol (described in control−spec.txt in torspec). Note: unless you also specify one or more of **HashedControlPassword** or **CookieAuthentication**, setting this option will cause Tor to allow any process on the local host to control it. (Setting both authentication methods means either method is sufficient to authenticate to Tor.) This option is required for many Tor controllers; most use the value of 9051. If a unix domain socket is used, you may quote the path using standard C escape sequences. You can specify this directive multiple times, to bind to multiple address/port pairs. Set it to "auto" to have Tor pick a port for you. (Default: 0)

Recognized flags are:

GroupWritable
Unix domain sockets only: makes the socket get created as group−writable.

WorldWritable
Unix domain sockets only: makes the socket get created as world−writable.

RelaxDirModeCheck
Unix domain sockets only: Do not insist that the directory that holds the socket be read−restricted.

ControlPortFileGroupReadable 0|**1**
If this option is set to 0, don’t allow the filesystem group to read the control port file. If the option is set to 1, make the control port file readable by the default GID. (Default: 0)

ControlPortWriteToFile *Path*
If set, Tor writes the address and port of any control port it opens to this address. Usable by controllers to learn the actual control port when ControlPort is set to "auto".

ControlSocket *Path*
Like ControlPort, but listens on a Unix domain socket, rather than a TCP socket. *0* disables ControlSocket. (Unix and Unix−like systems only.) (Default: 0)

ControlSocketsGroupWritable 0|**1**
If this option is set to 0, don’t allow the filesystem group to read and write unix sockets (e.g. ControlSocket). If the option is set to 1, make the control socket readable and writable by the default GID. (Default: 0)

CookieAuthentication 0|**1**
If this option is set to 1, allow connections on the control port when the connecting process knows the contents of a file named "control_auth_cookie", which Tor will create in its data directory. This authentication method should only be used on systems with good filesystem security. (Default: 0)

CookieAuthFile *Path*
If set, this option overrides the default location and file name for Tor’s cookie file. (See CookieAuthentication.)

CookieAuthFileGroupReadable 0|**1**
If this option is set to 0, don’t allow the filesystem group to read the cookie file. If the option is set to 1, make the cookie file readable by the default GID. [Making the file readable by other groups is not yet implemented; let us know if you need this for some reason.] (Default: 0)

CountPrivateBandwidth 0|**1**
If this option is set, then Tor’s rate−limiting applies not only to remote connections, but also to connections to private addresses like 127.0.0.1 or 10.0.0.1. This is mostly useful for debugging

rate−limiting. (Default: 0)

DataDirectory *DIR*
Store working data in DIR. Can not be changed while tor is running. (Default: ~/.tor if your home directory is not /; otherwise, /var/lib/tor. On Windows, the default is your ApplicationData folder.)

DataDirectoryGroupReadable 0|**1**
If this option is set to 0, don't allow the filesystem group to read the DataDirectory. If the option is set to 1, make the DataDirectory readable by the default GID. (Default: 0)

DirAuthority [*nickname*] [**flags**] *ipv4address*:*dirport fingerprint*
Use a nonstandard authoritative directory server at the provided address and port, with the specified key fingerprint. This option can be repeated many times, for multiple authoritative directory servers. Flags are separated by spaces, and determine what kind of an authority this directory is. By default, an authority is not authoritative for any directory style or version unless an appropriate flag is given.

Tor will use this authority as a bridge authoritative directory if the "bridge" flag is set. If a flag "orport=**orport**" is given, Tor will use the given port when opening encrypted tunnels to the dirserver. If a flag "weight=**num**" is given, then the directory server is chosen randomly with probability proportional to that weight (default 1.0). If a flag "v3ident=**fp**" is given, the dirserver is a v3 directory authority whose v3 long−term signing key has the fingerprint **fp**. Lastly, if an "ipv6=**[***ipv6address***]**:*orport*" flag is present, then the directory authority is listening for IPv6 connections on the indicated IPv6 address and OR Port.

Tor will contact the authority at *ipv4address* to download directory documents. Clients always use the ORPort. Relays usually use the DirPort, but will use the ORPort in some circumstances. If an IPv6 ORPort is supplied, clients will also download directory documents at the IPv6 ORPort, if they are configured to use IPv6.

If no **DirAuthority** line is given, Tor will use the default directory authorities. NOTE: this option is intended for setting up a private Tor network with its own directory authorities. If you use it, you will be distinguishable from other users, because you won't believe the same authorities they do.

DirAuthorityFallbackRate *NUM*
When configured to use both directory authorities and fallback directories, the directory authorities also work as fallbacks. They are chosen with their regular weights, multiplied by this number, which should be 1.0 or less. The default is less than 1, to reduce load on authorities. (Default: 0.1)

DisableAllSwap 0|**1**
If set to 1, Tor will attempt to lock all current and future memory pages, so that memory cannot be paged out. Windows, OS X and Solaris are currently not supported. We believe that this feature works on modern Gnu/Linux distributions, and that it should work on *BSD systems (untested). This option requires that you start your Tor as root, and you should use the **User** option to properly reduce Tor's privileges. Can not be changed while tor is running. (Default: 0)

DisableDebuggerAttachment 0|**1**
If set to 1, Tor will attempt to prevent basic debugging attachment attempts by other processes. This may also keep Tor from generating core files if it crashes. It has no impact for users who wish to attach if they have CAP_SYS_PTRACE or if they are root. We believe that this feature works on modern Gnu/Linux distributions, and that it may also work on *BSD systems (untested). Some modern Gnu/Linux systems such as Ubuntu have the kernel.yama.ptrace_scope sysctl and by default enable it as an attempt to limit the PTRACE scope for all user processes by default. This feature will attempt to limit the PTRACE scope for Tor specifically − it will not attempt to alter the system wide ptrace scope as it may not even exist. If you wish to attach to Tor with a debugger such as gdb or strace you will want to set this to 0 for the duration of your debugging. Normal users should leave it on. Disabling this

option while Tor is running is prohibited. (Default: 1)

DisableNetwork 0|1
When this option is set, we don't listen for or accept any connections other than controller connections, and we close (and don't reattempt) any outbound connections. Controllers sometimes use this option to avoid using the network until Tor is fully configured. Tor will make still certain network−related calls (like DNS lookups) as a part of its configuration process, even if DisableNetwork is set. (Default: 0)

ExtendByEd25519ID 0|1|auto
If this option is set to 1, we always try to include a relay's Ed25519 ID when telling the preceding relay in a circuit to extend to it. If this option is set to 0, we never include Ed25519 IDs when extending circuits. If the option is set to "auto", we obey a parameter in the consensus document. (Default: auto)

ExtORPort [*address***:**]*port*|**auto**
Open this port to listen for Extended ORPort connections from your pluggable transports.

(Default: **DataDirectory**/extended_orport_auth_cookie)

ExtORPortCookieAuthFile *Path*
If set, this option overrides the default location and file name for the Extended ORPort's cookie file — the cookie file is needed for pluggable transports to communicate through the Extended ORPort.

ExtORPortCookieAuthFileGroupReadable 0|1
If this option is set to 0, don't allow the filesystem group to read the Extended OR Port cookie file. If the option is set to 1, make the cookie file readable by the default GID. [Making the file readable by other groups is not yet implemented; let us know if you need this for some reason.] (Default: 0)

FallbackDir *ipv4address*:*dirport* orport=*orport* id=*fingerprint* [weight=*num*] [ipv6=**[***ipv6address***]**:*orport*]
When tor is unable to connect to any directory cache for directory info (usually because it doesn't know about any yet) it tries a hard−coded directory. Relays try one directory authority at a time. Clients try multiple directory authorities and FallbackDirs, to avoid hangs on startup if a hard−coded directory is down. Clients wait for a few seconds between each attempt, and retry FallbackDirs more often than directory authorities, to reduce the load on the directory authorities.

FallbackDirs should be stable relays with stable IP addresses, ports, and identity keys. They must have a DirPort.

By default, the directory authorities are also FallbackDirs. Specifying a FallbackDir replaces Tor's default hard−coded FallbackDirs (if any). (See DirAuthority for an explanation of each flag.)

FetchDirInfoEarly 0|1
If set to 1, Tor will always fetch directory information like other directory caches, even if you don't meet the normal criteria for fetching early. Normal users should leave it off. (Default: 0)

FetchDirInfoExtraEarly 0|1
If set to 1, Tor will fetch directory information before other directory caches. It will attempt to download directory information closer to the start of the consensus period. Normal users should leave it off. (Default: 0)

FetchHidServDescriptors 0|1
If set to 0, Tor will never fetch any hidden service descriptors from the rendezvous directories. This option is only useful if you're using a Tor controller that handles hidden service fetches for you. (Default: 1)

FetchServerDescriptors 0|1
If set to 0, Tor will never fetch any network status summaries or server descriptors from the directory

servers. This option is only useful if you're using a Tor controller that handles directory fetches for you. (Default: 1)

FetchUselessDescriptors 0|1
If set to 1, Tor will fetch every consensus flavor, and all server descriptors and authority certificates referenced by those consensuses, except for extra info descriptors. When this option is 1, Tor will also keep fetching descriptors, even when idle. If set to 0, Tor will avoid fetching useless descriptors: flavors that it is not using to build circuits, and authority certificates it does not trust. When Tor hasn't built any application circuits, it will go idle, and stop fetching descriptors. This option is useful if you're using a tor client with an external parser that uses a full consensus. This option fetches all documents except extrainfo descriptors, **DirCache** fetches and serves all documents except extrainfo descriptors, **DownloadExtraInfo*** fetches extrainfo documents, and serves them if **DirCache** is on, and **UseMicrodescriptors** changes the flavor of consensuses and descriptors that is fetched and used for building circuits. (Default: 0)

HardwareAccel 0|1
If non−zero, try to use built−in (static) crypto hardware acceleration when available. Can not be changed while tor is running. (Default: 0)

HashedControlPassword *hashed_password*
Allow connections on the control port if they present the password whose one−way hash is *hashed_password*. You can compute the hash of a password by running "tor −−hash−password *password*". You can provide several acceptable passwords by using more than one HashedControlPassword line.

HTTPProxy *host*[:*port*]
Tor will make all its directory requests through this host:port (or host:80 if port is not specified), rather than connecting directly to any directory servers. (DEPRECATED: As of 0.3.1.0−alpha you should use HTTPSProxy.)

HTTPProxyAuthenticator *username:password*
If defined, Tor will use this username:password for Basic HTTP proxy authentication, as in RFC 2617. This is currently the only form of HTTP proxy authentication that Tor supports; feel free to submit a patch if you want it to support others. (DEPRECATED: As of 0.3.1.0−alpha you should use HTTPSProxyAuthenticator.)

HTTPSProxy *host*[:*port*]
Tor will make all its OR (SSL) connections through this host:port (or host:443 if port is not specified), via HTTP CONNECT rather than connecting directly to servers. You may want to set **FascistFirewall** to restrict the set of ports you might try to connect to, if your HTTPS proxy only allows connecting to certain ports.

HTTPSProxyAuthenticator *username:password*
If defined, Tor will use this username:password for Basic HTTPS proxy authentication, as in RFC 2617. This is currently the only form of HTTPS proxy authentication that Tor supports; feel free to submit a patch if you want it to support others.

KeepalivePeriod *NUM*
To keep firewalls from expiring connections, send a padding keepalive cell every NUM seconds on open connections that are in use. (Default: 5 minutes)

KeepBindCapabilities 0|1|auto
On Linux, when we are started as root and we switch our identity using the **User** option, the **KeepBindCapabilities** option tells us whether to try to retain our ability to bind to low ports. If this value is 1, we try to keep the capability; if it is 0 we do not; and if it is **auto**, we keep the capability only if we are configured to listen on a low port. Can not be changed while tor is running. (Default: auto.)

Log *minSeverity*[−*maxSeverity*] **stderr|stdout|syslog**
Send all messages between *minSeverity* and *maxSeverity* to the standard output stream, the standard

error stream, or to the system log. (The "syslog" value is only supported on Unix.) Recognized severity levels are debug, info, notice, warn, and err. We advise using "notice" in most cases, since anything more verbose may provide sensitive information to an attacker who obtains the logs. If only one severity level is given, all messages of that level or higher will be sent to the listed destination.

Some low−level logs may be sent from signal handlers, so their destination logs must be signal−safe. These low−level logs include backtraces, logging function errors, and errors in code called by logging functions. Signal−safe logs are always sent to stderr or stdout. They are also sent to a limited number of log files that are configured to log messages at error severity from the bug or general domains. They are never sent as syslogs, control port log events, or to any API−based log destinations.

Log *minSeverity*[−*maxSeverity*] **file** *FILENAME*
As above, but send log messages to the listed filename. The "Log" option may appear more than once in a configuration file. Messages are sent to all the logs that match their severity level.

Log [*domain,...***]***minSeverity*[−*maxSeverity*] ... **file** *FILENAME*

Log [*domain,...***]***minSeverity*[−*maxSeverity*] ... **stderr**|**stdout**|**syslog**
As above, but select messages by range of log severity *and* by a set of "logging domains". Each logging domain corresponds to an area of functionality inside Tor. You can specify any number of severity ranges for a single log statement, each of them prefixed by a comma−separated list of logging domains. You can prefix a domain with ˜ to indicate negation, and use * to indicate "all domains". If you specify a severity range without a list of domains, it matches all domains.

This is an advanced feature which is most useful for debugging one or two of Tor's subsystems at a time.

The currently recognized domains are: general, crypto, net, config, fs, protocol, mm, http, app, control, circ, rend, bug, dir, dirserv, or, edge, acct, hist, handshake, heartbeat, channel, sched, guard, consdiff, dos, process, pt, btrack, and mesg. Domain names are case−insensitive.

For example, "Log [handshake]debug [˜net,˜mm]info notice stdout" sends to stdout: all handshake messages of any severity, all info−and−higher messages from domains other than networking and memory management, and all messages of severity notice or higher.

LogMessageDomains 0|**1**
If 1, Tor includes message domains with each log message. Every log message currently has at least one domain; most currently have exactly one. This doesn't affect controller log messages. (Default: 0)

LogTimeGranularity *NUM*
Set the resolution of timestamps in Tor's logs to NUM milliseconds. NUM must be positive and either a divisor or a multiple of 1 second. Note that this option only controls the granularity written by Tor to a file or console log. Tor does not (for example) "batch up" log messages to affect times logged by a controller, times attached to syslog messages, or the mtime fields on log files. (Default: 1 second)

MaxAdvertisedBandwidth *N* **bytes**|**KBytes**|**MBytes**|**GBytes**|**TBytes**|**KBits**|**MBits**|**GBits**|**TBits**
If set, we will not advertise more than this amount of bandwidth for our BandwidthRate. Server operators who want to reduce the number of clients who ask to build circuits through them (since this is proportional to advertised bandwidth rate) can thus reduce the CPU demands on their server without impacting network performance.

MaxUnparseableDescSizeToLog *N* **bytes**|**KBytes**|**MBytes**|**GBytes**|**TBytes**
Unparseable descriptors (e.g. for votes, consensuses, routers) are logged in separate files by hash, up to the specified size in total. Note that only files logged during the lifetime of this Tor process count

toward the total; this is intended to be used to debug problems without opening live servers to resource exhaustion attacks. (Default: 10 MBytes)

MetricsPort [*address***:**]*port* [*format*]
WARNING: Before enabling this, it is important to understand that exposing tor metrics publicly is dangerous to the Tor network users. Please take extra precaution and care when opening this port. Set a very strict access policy with MetricsPortPolicy and consider using your operating systems firewall features for defense in depth.

We recommend, for the prometheus *format*, that the only address that can access this port should be the Prometheus server itself. Remember that the connection is unencrypted (HTTP) hence consider using a tool like stunnel to secure the link from this port to the server.

If set, open this port to listen for an HTTP GET request to "/metrics". Upon a request, the collected metrics in the the tor instance are formatted for the given format and then sent back. If this is set, MetricsPortPolicy must be defined else every request will be rejected.

Supported format is "prometheus" which is also the default if not set. The Prometheus data model can be found here: https://prometheus.io/docs/concepts/data_model/

The tor metrics are constantly collected and they solely consists of counters. Thus, asking for those metrics is very lightweight on the tor process. (Default: None)

As an example, here only 5.6.7.8 will be allowed to connect:

MetricsPort 1.2.3.4:9035
MetricsPortPolicy accept 5.6.7.8

MetricsPortPolicy *policy*,*policy*,...
Set an entrance policy for the **MetricsPort**, to limit who can access it. The policies have the same form as exit policies below, except that port specifiers are ignored. For multiple entries, this line can be used multiple times. It is a reject all by default policy. (Default: None)

Please, keep in mind here that if the server collecting metrics on the MetricsPort is behind a NAT, then everything behind it can access it. This is similar for the case of allowing localhost, every users on the server will be able to access it. Again, strongly consider using a tool like stunnel to secure the link or to strengthen access control.

NoExec 0|1
If this option is set to 1, then Tor will never launch another executable, regardless of the settings of ClientTransportPlugin or ServerTransportPlugin. Once this option has been set to 1, it cannot be set back to 0 without restarting Tor. (Default: 0)

OutboundBindAddress *IP*
Make all outbound connections originate from the IP address specified. This is only useful when you have multiple network interfaces, and you want all of Tor's outgoing connections to use a single one. This option may be used twice, once with an IPv4 address and once with an IPv6 address. IPv6 addresses should be wrapped in square brackets. This setting will be ignored for connections to the loopback addresses (127.0.0.0/8 and ::1), and is not used for DNS requests as well.

OutboundBindAddressExit *IP*
Make all outbound exit connections originate from the IP address specified. This option overrides **OutboundBindAddress** for the same IP version. This option may be used twice, once with an IPv4 address and once with an IPv6 address. IPv6 addresses should be wrapped in square brackets. This setting will be ignored for connections to the loopback addresses (127.0.0.0/8 and ::1).

OutboundBindAddressOR *IP*
Make all outbound non−exit (relay and other) connections originate from the IP address specified.

This option overrides **OutboundBindAddress** for the same IP version. This option may be used twice, once with an IPv4 address and once with an IPv6 address. IPv6 addresses should be wrapped in square brackets. This setting will be ignored for connections to the loopback addresses (127.0.0.0/8 and ::1).

__OwningControllerProcess *PID*
Make Tor instance periodically check for presence of a controller process with given PID and terminate itself if this process is no longer alive. Polling interval is 15 seconds.

PerConnBWBurst *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
If this option is set manually, or via the "perconnbwburst" consensus field, Tor will use it for separate rate limiting for each connection from a non−relay. (Default: 0)

PerConnBWRate *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
If this option is set manually, or via the "perconnbwrate" consensus field, Tor will use it for separate rate limiting for each connection from a non−relay. (Default: 0)

OutboundBindAddressPT *IP*
Request that pluggable transports makes all outbound connections originate from the IP address specified. Because outgoing connections are handled by the pluggable transport itself, it is not possible for Tor to enforce whether the pluggable transport honors this option. This option overrides **OutboundBindAddress** for the same IP version. This option may be used twice, once with an IPv4 address and once with an IPv6 address. IPv6 addresses should be wrapped in square brackets. This setting will be ignored for connections to the loopback addresses (127.0.0.0/8 and ::1).

PidFile *FILE*
On startup, write our PID to FILE. On clean shutdown, remove FILE. Can not be changed while tor is running.

ProtocolWarnings 0|1
If 1, Tor will log with severity 'warn' various cases of other parties not following the Tor specification. Otherwise, they are logged with severity 'info'. (Default: 0)

RelayBandwidthBurst *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
If not 0, limit the maximum token bucket size (also known as the burst) for _relayed traffic_ to the given number of bytes in each direction. They do not include directory fetches by the relay (from authority or other relays), because that is considered "client" activity. (Default: 0)

RelayBandwidthRate *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
If not 0, a separate token bucket limits the average incoming bandwidth usage for _relayed traffic_ on this node to the specified number of bytes per second, and the average outgoing bandwidth usage to that same value. Relayed traffic currently is calculated to include answers to directory requests, but that may change in future versions. They do not include directory fetches by the relay (from authority or other relays), because that is considered "client" activity. (Default: 0)

RephistTrackTime *N* **seconds|minutes|hours|days|weeks**
Tells an authority, or other node tracking node reliability and history, that fine−grained information about nodes can be discarded when it hasn’t changed for a given amount of time. (Default: 24 hours)

RunAsDaemon 0|1
If 1, Tor forks and daemonizes to the background. This option has no effect on Windows; instead you should use the −−service command−line option. Can not be changed while tor is running. (Default: 0)

SafeLogging 0|1|relay
Tor can scrub potentially sensitive strings from log messages (e.g. addresses) by replacing them with the string [scrubbed]. This way logs can still be useful, but they don’t leave behind personally identifying information about what sites a user might have visited.

If this option is set to 0, Tor will not perform any scrubbing, if it is set to 1, all potentially sensitive strings are replaced. If it is set to relay, all log messages generated when acting as a relay are sanitized,

but all messages generated when acting as a client are not. Note: Tor may not heed this option when logging at log levels below Notice. (Default: 1)

Sandbox 0|1
If set to 1, Tor will run securely through the use of a syscall sandbox. Otherwise the sandbox will be disabled. The option only works on Linux−based operating systems, and only when Tor has been built with the libseccomp library. Note that this option may be incompatible with some versions of libc, and some kernel versions. This option can not be changed while tor is running.

When the **Sandbox** is 1, the following options can not be changed when tor is running: **Address**, **ConnLimit**, **CookieAuthFile**, **DirPortFrontPage**, **ExtORPortCookieAuthFile**, **Logs**, **ServerDNSResolvConfFile**, **ClientOnionAuthDir** (and any files in it won't reload on HUP signal).

Launching new Onion Services through the control port is not supported with current syscall sandboxing implementation.

Tor must remain in client or server mode (some changes to **ClientOnly** and **ORPort** are not allowed). Currently, if **Sandbox** is 1, **ControlPort** command "GETINFO address" will not work.

When using %include in the tor configuration files, reloading the tor configuration is not supported after adding new configuration files or directories.

(Default: 0)

Schedulers KIST|KISTLite|Vanilla
Specify the scheduler type that tor should use. The scheduler is responsible for moving data around within a Tor process. This is an ordered list by priority which means that the first value will be tried first and if unavailable, the second one is tried and so on. It is possible to change these values at runtime. This option mostly effects relays, and most operators should leave it set to its default value. (Default: KIST,KISTLite,Vanilla)

The possible scheduler types are:

KIST: Kernel−Informed Socket Transport. Tor will use TCP information from the kernel to make informed decisions regarding how much data to send and when to send it. KIST also handles traffic in batches (see KISTSchedRunInterval) in order to improve traffic prioritization decisions. As implemented, KIST will only work on Linux kernel version 2.6.39 or higher.

KISTLite: Same as KIST but without kernel support. Tor will use all the same mechanics as with KIST, including the batching, but its decisions regarding how much data to send will not be as good. KISTLite will work on all kernels and operating systems, and the majority of the benefits of KIST are still realized with KISTLite.

Vanilla: The scheduler that Tor used before KIST was implemented. It sends as much data as possible, as soon as possible. Vanilla will work on all kernels and operating systems.

KISTSchedRunInterval *NUM* **msec**
If KIST or KISTLite is used in the Schedulers option, this controls at which interval the scheduler tick

is. If the value is 0 msec, the value is taken from the consensus if possible else it will fallback to the default 10 msec. Maximum possible value is 100 msec. (Default: 0 msec)

KISTSockBufSizeFactor *NUM*
If KIST is used in Schedulers, this is a multiplier of the per−socket limit calculation of the KIST algorithm. (Default: 1.0)

Socks4Proxy *host*[:*port*]
Tor will make all OR connections through the SOCKS 4 proxy at host:port (or host:1080 if port is not specified).

Socks5Proxy *host*[:*port*]
Tor will make all OR connections through the SOCKS 5 proxy at host:port (or host:1080 if port is not specified).

Socks5ProxyUsername *username*

Socks5ProxyPassword *password*
If defined, authenticate to the SOCKS 5 server using username and password in accordance to RFC 1929. Both username and password must be between 1 and 255 characters.

SyslogIdentityTag *tag*
When logging to syslog, adds a tag to the syslog identity such that log entries are marked with "Tor−*tag*". Can not be changed while tor is running. (Default: none)

TCPProxy *protocol host*:*port*
Tor will use the given protocol to make all its OR (SSL) connections through a TCP proxy on host:port, rather than connecting directly to servers. You may want to set **FascistFirewall** to restrict the set of ports you might try to connect to, if your proxy only allows connecting to certain ports. There is no equivalent option for directory connections, because all Tor client versions that support this option download directory documents via OR connections.

The only protocol supported right now 'haproxy'. This option is only for clients. (Default: none) +

The HAProxy version 1 proxy protocol is described in detail at https://www.haproxy.org/download/1.8/doc/proxy−protocol.txt +

Both source IP address and source port will be set to zero.

TruncateLogFile 0|1
If 1, Tor will overwrite logs at startup and in response to a HUP signal, instead of appending to them. (Default: 0)

UnixSocksGroupWritable 0|1
If this option is set to 0, don’t allow the filesystem group to read and write unix sockets (e.g. SocksPort unix:). If the option is set to 1, make the Unix socket readable and writable by the default GID. (Default: 0)

UseDefaultFallbackDirs 0|1
Use Tor’s default hard−coded FallbackDirs (if any). (When a FallbackDir line is present, it replaces the hard−coded FallbackDirs, regardless of the value of UseDefaultFallbackDirs.) (Default: 1)

User *Username*
On startup, setuid to this user and setgid to their primary group. Can not be changed while tor is running.

CLIENT OPTIONS

The following options are useful only for clients (that is, if **SocksPort**, **HTTPTunnelPort**, **TransPort**, **DNSPort**, or **NATDPort** is non−zero):

AllowNonRFC953Hostnames 0|1
When this option is disabled, Tor blocks hostnames containing illegal characters (like @ and :) rather than sending them to an exit node to be resolved. This helps trap accidental attempts to resolve URLs and so on. (Default: 0)

AutomapHostsOnResolve 0|1
When this option is enabled, and we get a request to resolve an address that ends with one of the suffixes in **AutomapHostsSuffixes**, we map an unused virtual address to that address, and return the new virtual address. This is handy for making ".onion" addresses work with applications that resolve an address and then connect to it. (Default: 0)

AutomapHostsSuffixes *SUFFIX*,*SUFFIX*,...
A comma−separated list of suffixes to use with **AutomapHostsOnResolve**. The "." suffix is equivalent to "all addresses." (Default: .exit,.onion).

Bridge [*transport*] *IP*:*ORPort* [*fingerprint*]
When set along with UseBridges, instructs Tor to use the relay at "IP:ORPort" as a "bridge" relaying into the Tor network. If "fingerprint" is provided (using the same format as for DirAuthority), we will verify that the relay running at that location has the right fingerprint. We also use fingerprint to look up the bridge descriptor at the bridge authority, if it's provided and if UpdateBridgesFromAuthority is set too.

If "transport" is provided, it must match a ClientTransportPlugin line. We then use that pluggable transport's proxy to transfer data to the bridge, rather than connecting to the bridge directly. Some transports use a transport−specific method to work out the remote address to connect to. These transports typically ignore the "IP:ORPort" specified in the bridge line.

Tor passes any "key=val" settings to the pluggable transport proxy as per−connection arguments when connecting to the bridge. Consult the documentation of the pluggable transport for details of what arguments it supports.

CircuitPadding 0|1
If set to 0, Tor will not pad client circuits with additional cover traffic. Only clients may set this option. This option should be offered via the UI to mobile users for use where bandwidth may be expensive. If set to 1, padding will be negotiated as per the consensus and relay support (unlike ConnectionPadding, CircuitPadding cannot be force−enabled). (Default: 1)

ReducedCircuitPadding 0|1
If set to 1, Tor will only use circuit padding algorithms that have low overhead. Only clients may set this option. This option should be offered via the UI to mobile users for use where bandwidth may be expensive. (Default: 0)

ClientBootstrapConsensusAuthorityDownloadInitialDelay *N*
Initial delay in seconds for when clients should download consensuses from authorities if they are bootstrapping (that is, they don't have a usable, reasonably live consensus). Only used by clients fetching from a list of fallback directory mirrors. This schedule is advanced by (potentially concurrent) connection attempts, unlike other schedules, which are advanced by connection failures. (Default: 6)

ClientBootstrapConsensusAuthorityOnlyDownloadInitialDelay *N*
Initial delay in seconds for when clients should download consensuses from authorities if they are bootstrapping (that is, they don't have a usable, reasonably live consensus). Only used by clients which don't have or won't fetch from a list of fallback directory mirrors. This schedule is advanced by (potentially concurrent) connection attempts, unlike other schedules, which are advanced by connection failures. (Default: 0)

ClientBootstrapConsensusFallbackDownloadInitialDelay *N*
Initial delay in seconds for when clients should download consensuses from fallback directory mirrors

if they are bootstrapping (that is, they don't have a usable, reasonably live consensus). Only used by clients fetching from a list of fallback directory mirrors. This schedule is advanced by (potentially concurrent) connection attempts, unlike other schedules, which are advanced by connection failures. (Default: 0)

ClientBootstrapConsensusMaxInProgressTries *NUM*
Try this many simultaneous connections to download a consensus before waiting for one to complete, timeout, or error out. (Default: 3)

ClientDNSRejectInternalAddresses 0|1
If true, Tor does not believe any anonymously retrieved DNS answer that tells it that an address resolves to an internal address (like 127.0.0.1 or 192.168.0.1). This option prevents certain browser−based attacks; it is not allowed to be set on the default network. (Default: 1)

ClientOnionAuthDir *path*
Path to the directory containing v3 hidden service authorization files. Each file is for a single onion address, and the files MUST have the suffix ".auth_private" (i.e. "bob_onion.auth_private"). The content format MUST be:

<onion−address>:descriptor:x25519:<base32−encoded−privkey>

The <onion−address> MUST NOT have the ".onion" suffix. The <base32−encoded−privkey> is the base32 representation of the raw key bytes only (32 bytes for x25519). See Appendix G in the rend−spec−v3.txt file of torspec for more information.

ClientOnly 0|1
If set to 1, Tor will not run as a relay or serve directory requests, even if the ORPort, ExtORPort, or DirPort options are set. (This config option is mostly unnecessary: we added it back when we were considering having Tor clients auto−promote themselves to being relays if they were stable and fast enough. The current behavior is simply that Tor is a client unless ORPort, ExtORPort, or DirPort are configured.) (Default: 0)

ClientPreferIPv6DirPort 0|1|auto
If this option is set to 1, Tor prefers a directory port with an IPv6 address over one with IPv4, for direct connections, if a given directory server has both. (Tor also prefers an IPv6 DirPort if IPv4Client is set to 0.) If this option is set to auto, clients prefer IPv4. Other things may influence the choice. This option breaks a tie to the favor of IPv6. (Default: auto) (DEPRECATED: This option has had no effect for some time.)

ClientPreferIPv6ORPort 0|1|auto
If this option is set to 1, Tor prefers an OR port with an IPv6 address over one with IPv4 if a given entry node has both. (Tor also prefers an IPv6 ORPort if IPv4Client is set to 0.) If this option is set to auto, Tor bridge clients prefer the configured bridge address, and other clients prefer IPv4. Other things may influence the choice. This option breaks a tie to the favor of IPv6. (Default: auto)

ClientRejectInternalAddresses 0|1
If true, Tor does not try to fulfill requests to connect to an internal address (like 127.0.0.1 or 192.168.0.1) *unless an exit node is specifically requested* (for example, via a .exit hostname, or a controller request). If true, multicast DNS hostnames for machines on the local network (of the form *.local) are also rejected. (Default: 1)

ClientUseIPv4 0|1
If this option is set to 0, Tor will avoid connecting to directory servers and entry nodes over IPv4. Note that clients with an IPv4 address in a **Bridge**, proxy, or pluggable transport line will try connecting over IPv4 even if **ClientUseIPv4** is set to 0. (Default: 1)

ClientUseIPv6 0|1
If this option is set to 1, Tor might connect to directory servers or entry nodes over IPv6. For IPv6 only hosts, you need to also set **ClientUseIPv4** to 0 to disable IPv4. Note that clients configured with an IPv6 address in a **Bridge**, proxy, or pluggable transportline will try connecting over IPv6 even if

ClientUseIPv6 is set to 0. (Default: 0)

ConnectionPadding 0|1|auto
This option governs Tor's use of padding to defend against some forms of traffic analysis. If it is set to *auto*, Tor will send padding only if both the client and the relay support it. If it is set to 0, Tor will not send any padding cells. If it is set to 1, Tor will still send padding for client connections regardless of relay support. Only clients may set this option. This option should be offered via the UI to mobile users for use where bandwidth may be expensive. (Default: auto)

ReducedConnectionPadding 0|1
If set to 1, Tor will not not hold OR connections open for very long, and will send less padding on these connections. Only clients may set this option. This option should be offered via the UI to mobile users for use where bandwidth may be expensive. (Default: 0)

DNSPort [*address***:**]*port*|**auto** [*isolation flags*]
If non−zero, open this port to listen for UDP DNS requests, and resolve them anonymously. This port only handles A, AAAA, and PTR requests−−−it doesn't handle arbitrary DNS request types. Set the port to "auto" to have Tor pick a port for you. This directive can be specified multiple times to bind to multiple addresses/ports. See SocksPort for an explanation of isolation flags. (Default: 0)

DownloadExtraInfo 0|1
If true, Tor downloads and caches "extra−info" documents. These documents contain information about servers other than the information in their regular server descriptors. Tor does not use this information for anything itself; to save bandwidth, leave this option turned off. (Default: 0)

EnforceDistinctSubnets 0|1
If 1, Tor will not put two servers whose IP addresses are "too close" on the same circuit. Currently, two addresses are "too close" if they lie in the same /16 range. (Default: 1)

FascistFirewall 0|1
If 1, Tor will only create outgoing connections to ORs running on ports that your firewall allows (defaults to 80 and 443; see FirewallPorts). This will allow you to run Tor as a client behind a firewall with restrictive policies, but will not allow you to run as a server behind such a firewall. If you prefer more fine−grained control, use ReachableAddresses instead.

FirewallPorts *PORTS*
A list of ports that your firewall allows you to connect to. Only used when **FascistFirewall** is set. This option is deprecated; use ReachableAddresses instead. (Default: 80, 443)

HTTPTunnelPort [*address***:**]*port*|**auto** [*isolation flags*]
Open this port to listen for proxy connections using the "HTTP CONNECT" protocol instead of SOCKS. Set this to 0 if you don't want to allow "HTTP CONNECT" connections. Set the port to "auto" to have Tor pick a port for you. This directive can be specified multiple times to bind to multiple addresses/ports. If multiple entries of this option are present in your configuration file, Tor will perform stream isolation between listeners by default. See SocksPort for an explanation of isolation flags. (Default: 0)

LongLivedPorts *PORTS*
A list of ports for services that tend to have long−running connections (e.g. chat and interactive shells). Circuits for streams that use these ports will contain only high−uptime nodes, to reduce the chance that a node will go down before the stream is finished. Note that the list is also honored for circuits (both client and service side) involving hidden services whose virtual port is in this list. (Default: 21, 22, 706, 1863, 5050, 5190, 5222, 5223, 6523, 6667, 6697, 8300)

MapAddress *address newaddress*
When a request for address arrives to Tor, it will transform to newaddress before processing it. For example, if you always want connections to www.example.com to exit via *torserver* (where *torserver* is the fingerprint of the server), use "MapAddress www.example.com www.example.com.torserver.exit". If the value is prefixed with a "*.", matches an entire domain. For example, if you always want connections to example.com and any if its subdomains to exit via

torserver (where *torserver* is the fingerprint of the server), use "MapAddress *.example.com *.example.com.torserver.exit". (Note the leading "*." in each part of the directive.) You can also redirect all subdomains of a domain to a single address. For example, "MapAddress *.example.com www.example.com". If the specified exit is not available, or the exit can not connect to the site, Tor will fail any connections to the mapped address.+

NOTES:

1. When evaluating MapAddress expressions Tor stops when it hits the most recently added expression that matches the requested address. So if you have the following in your torrc, www.torproject.org will map to 198.51.100.1:

   MapAddress www.torproject.org 192.0.2.1
   MapAddress www.torproject.org 198.51.100.1

2. Tor evaluates the MapAddress configuration until it finds no matches. So if you have the following in your torrc, www.torproject.org will map to 203.0.113.1:

   MapAddress 198.51.100.1 203.0.113.1
   MapAddress www.torproject.org 198.51.100.1

3. The following MapAddress expression is invalid (and will be ignored) because you cannot map from a specific address to a wildcard address:

   MapAddress www.torproject.org *.torproject.org.torserver.exit

4. Using a wildcard to match only part of a string (as in *ample.com) is also invalid.
5. Tor maps hostnames and IP addresses separately. If you MapAddress a DNS name, but use an IP address to connect, then Tor will ignore the DNS name mapping.
6. MapAddress does not apply to redirects in the application protocol. For example, HTTP redirects and alt−svc headers will ignore mappings for the original address. You can use a wildcard mapping to handle redirects within the same site.

MaxCircuitDirtiness *NUM*
Feel free to reuse a circuit that was first used at most NUM seconds ago, but never attach a new stream to a circuit that is too old. For hidden services, this applies to the *last* time a circuit was used, not the first. Circuits with streams constructed with SOCKS authentication via SocksPorts that have **KeepAliveIsolateSOCKSAuth** also remain alive for MaxCircuitDirtiness seconds after carrying the last such stream. (Default: 10 minutes)

MaxClientCircuitsPending *NUM*
Do not allow more than NUM circuits to be pending at a time for handling client streams. A circuit is pending if we have begun constructing it, but it has not yet been completely constructed. (Default: 32)

NATDPort [*address***:**]*port*|**auto** [*isolation flags*]
Open this port to listen for connections from old versions of ipfw (as included in old versions of FreeBSD, etc) using the NATD protocol. Use 0 if you don’t want to allow NATD connections. Set the port to "auto" to have Tor pick a port for you. This directive can be specified multiple times to bind to multiple addresses/ports. If multiple entries of this option are present in your configuration file, Tor will perform stream isolation between listeners by default. See SocksPort for an explanation of isolation flags.

This option is only for people who cannot use TransPort. (Default: 0)

NewCircuitPeriod *NUM*
Every NUM seconds consider whether to build a new circuit. (Default: 30 seconds)

PathBiasCircThreshold *NUM*

PathBiasDropGuards *NUM*

PathBiasExtremeRate *NUM*

PathBiasNoticeRate *NUM*

PathBiasWarnRate *NUM*

PathBiasScaleThreshold *NUM*
These options override the default behavior of Tor's (**currently experimental**) path bias detection algorithm. To try to find broken or misbehaving guard nodes, Tor looks for nodes where more than a certain fraction of circuits through that guard fail to get built.

The PathBiasCircThreshold option controls how many circuits we need to build through a guard before we make these checks. The PathBiasNoticeRate, PathBiasWarnRate and PathBiasExtremeRate options control what fraction of circuits must succeed through a guard so we won't write log messages. If less than PathBiasExtremeRate circuits succeed **and** PathBiasDropGuards is set to 1, we disable use of that guard.

When we have seen more than PathBiasScaleThreshold circuits through a guard, we scale our observations by 0.5 (governed by the consensus) so that new observations don't get swamped by old ones.

By default, or if a negative value is provided for one of these options, Tor uses reasonable defaults from the networkstatus consensus document. If no defaults are available there, these options default to 150, .70, .50, .30, 0, and 300 respectively.

PathBiasUseThreshold *NUM*

PathBiasNoticeUseRate *NUM*

PathBiasExtremeUseRate *NUM*

PathBiasScaleUseThreshold *NUM*
Similar to the above options, these options override the default behavior of Tor's (**currently experimental**) path use bias detection algorithm.

Where as the path bias parameters govern thresholds for successfully building circuits, these four path use bias parameters govern thresholds only for circuit usage. Circuits which receive no stream usage are not counted by this detection algorithm. A used circuit is considered successful if it is capable of carrying streams or otherwise receiving well−formed responses to RELAY cells.

By default, or if a negative value is provided for one of these options, Tor uses reasonable defaults from the networkstatus consensus document. If no defaults are available there, these options default to 20, .80, .60, and 100, respectively.

PathsNeededToBuildCircuits *NUM*
Tor clients don't build circuits for user traffic until they know about enough of the network so that they could potentially construct enough of the possible paths through the network. If this option is set to a

fraction between 0.25 and 0.95, Tor won't build circuits until it has enough descriptors or microdescriptors to construct that fraction of possible paths. Note that setting this option too low can make your Tor client less anonymous, and setting it too high can prevent your Tor client from bootstrapping. If this option is negative, Tor will use a default value chosen by the directory authorities. If the directory authorities do not choose a value, Tor will default to 0.6. (Default: −1)

ReachableAddresses *IP*[/*MASK*][:*PORT*]...
A comma−separated list of IP addresses and ports that your firewall allows you to connect to. The format is as for the addresses in ExitPolicy, except that "accept" is understood unless "reject" is explicitly provided. For example, 'ReachableAddresses 99.0.0.0/8, reject 18.0.0.0/8:80, accept *:80' means that your firewall allows connections to everything inside net 99, rejects port 80 connections to net 18, and accepts connections to port 80 otherwise. (Default: 'accept *:*'.)

ReachableDirAddresses *IP*[/*MASK*][:*PORT*]...
Like **ReachableAddresses**, a list of addresses and ports. Tor will obey these restrictions when fetching directory information, using standard HTTP GET requests. If not set explicitly then the value of **ReachableAddresses** is used. If **HTTPProxy** is set then these connections will go through that proxy. (DEPRECATED: This option has had no effect for some time.)

ReachableORAddresses *IP*[/*MASK*][:*PORT*]...
Like **ReachableAddresses**, a list of addresses and ports. Tor will obey these restrictions when connecting to Onion Routers, using TLS/SSL. If not set explicitly then the value of **ReachableAddresses** is used. If **HTTPSProxy** is set then these connections will go through that proxy.

The separation between **ReachableORAddresses** and **ReachableDirAddresses** is only interesting when you are connecting through proxies (see HTTPProxy and HTTPSProxy). Most proxies limit TLS connections (which Tor uses to connect to Onion Routers) to port 443, and some limit HTTP GET requests (which Tor uses for fetching directory information) to port 80.

SafeSocks 0|**1**
When this option is enabled, Tor will reject application connections that use unsafe variants of the socks protocol — ones that only provide an IP address, meaning the application is doing a DNS resolve first. Specifically, these are socks4 and socks5 when not doing remote DNS. (Default: 0)

TestSocks 0|**1**
When this option is enabled, Tor will make a notice−level log entry for each connection to the Socks port indicating whether the request used a safe socks protocol or an unsafe one (see SafeSocks). This helps to determine whether an application using Tor is possibly leaking DNS requests. (Default: 0)

WarnPlaintextPorts *port*,*port*,...
Tells Tor to issue a warnings whenever the user tries to make an anonymous connection to one of these ports. This option is designed to alert users to services that risk sending passwords in the clear. (Default: 23,109,110,143)

RejectPlaintextPorts *port*,*port*,...
Like WarnPlaintextPorts, but instead of warning about risky port uses, Tor will instead refuse to make the connection. (Default: None)

SocksPolicy *policy*,*policy*,...
Set an entrance policy for this server, to limit who can connect to the SocksPort and DNSPort ports. The policies have the same form as exit policies below, except that port specifiers are ignored. Any address not matched by some entry in the policy is accepted.

SocksPort [*address***:**]*port*|**unix:***path*|**auto** [*flags*] [*isolation flags*]
Open this port to listen for connections from SOCKS−speaking applications. Set this to 0 if you don't want to allow application connections via SOCKS. Set it to "auto" to have Tor pick a port for you. This directive can be specified multiple times to bind to multiple addresses/ports. If a unix domain socket is used, you may quote the path using standard C escape sequences. Most flags are off by default, except

where specified. Flags that are on by default can be disabled by putting "No" before the flag name. (Default: 9050)

NOTE: Although this option allows you to specify an IP address other than localhost, you should do so only with extreme caution. The SOCKS protocol is unencrypted and (as we use it) unauthenticated, so exposing it in this way could leak your information to anybody watching your network, and allow anybody to use your computer as an open proxy.

If multiple entries of this option are present in your configuration file, Tor will perform stream isolation between listeners by default. The *isolation flags* arguments give Tor rules for which streams received on this SocksPort are allowed to share circuits with one another. Recognized isolation flags are:

IsolateClientAddr
Don’t share circuits with streams from a different client address. (On by default and strongly recommended when supported; you can disable it with **NoIsolateClientAddr**. Unsupported and force−disabled when using Unix domain sockets.)

IsolateSOCKSAuth
Don’t share circuits with streams for which different SOCKS authentication was provided. (For HTTPTunnelPort connections, this option looks at the Proxy−Authorization and X−Tor−Stream−Isolation headers. On by default; you can disable it with **NoIsolateSOCKSAuth**.)

IsolateClientProtocol
Don’t share circuits with streams using a different protocol. (SOCKS 4, SOCKS 5, HTTPTunnelPort connections, TransPort connections, NATDPort connections, and DNSPort requests are all considered to be different protocols.)

IsolateDestPort
Don’t share circuits with streams targeting a different destination port.

IsolateDestAddr
Don’t share circuits with streams targeting a different destination address.

KeepAliveIsolateSOCKSAuth
If **IsolateSOCKSAuth** is enabled, keep alive circuits while they have at least one stream with SOCKS authentication active. After such a circuit is idle for more than MaxCircuitDirtiness seconds, it can be closed.

SessionGroup=*INT*
If no other isolation rules would prevent it, allow streams on this port to share circuits with streams from every other port with the same session group. (By default, streams received on different SocksPorts, TransPorts, etc are always isolated from one another. This option overrides that behavior.)

Other recognized *flags* for a SocksPort are:

NoIPv4Traffic
Tell exits to not connect to IPv4 addresses in response to SOCKS requests on this connection.

IPv6Traffic
Tell exits to allow IPv6 addresses in response to SOCKS requests on this connection, so long as SOCKS5 is in use. (SOCKS4 can’t handle IPv6.)

PreferIPv6
Tells exits that, if a host has both an IPv4 and an IPv6 address, we would prefer to connect to it via IPv6. (IPv4 is the default.)

NoDNSRequest

Do not ask exits to resolve DNS addresses in SOCKS5 requests. Tor will connect to IPv4 addresses, IPv6 addresses (if IPv6Traffic is set) and .onion addresses.

NoOnionTraffic
Do not connect to .onion addresses in SOCKS5 requests.

OnionTrafficOnly
Tell the tor client to only connect to .onion addresses in response to SOCKS5 requests on this connection. This is equivalent to NoDNSRequest, NoIPv4Traffic, NoIPv6Traffic. The corresponding NoOnionTrafficOnly flag is not supported.

CacheIPv4DNS
Tells the client to remember IPv4 DNS answers we receive from exit nodes via this connection.

CacheIPv6DNS
Tells the client to remember IPv6 DNS answers we receive from exit nodes via this connection.

GroupWritable
Unix domain sockets only: makes the socket get created as group−writable.

WorldWritable
Unix domain sockets only: makes the socket get created as world−writable.

CacheDNS
Tells the client to remember all DNS answers we receive from exit nodes via this connection.

UseIPv4Cache
Tells the client to use any cached IPv4 DNS answers we have when making requests via this connection. (NOTE: This option, or UseIPv6Cache or UseDNSCache, can harm your anonymity, and probably won’t help performance as much as you might expect. Use with care!)

UseIPv6Cache
Tells the client to use any cached IPv6 DNS answers we have when making requests via this connection.

UseDNSCache
Tells the client to use any cached DNS answers we have when making requests via this connection.

NoPreferIPv6Automap
When serving a hostname lookup request on this port that should get automapped (according to AutomapHostsOnResolve), if we could return either an IPv4 or an IPv6 answer, prefer an IPv4 answer. (Tor prefers IPv6 by default.)

PreferSOCKSNoAuth
Ordinarily, when an application offers both "username/password authentication" and "no authentication" to Tor via SOCKS5, Tor selects username/password authentication so that IsolateSOCKSAuth can work. This can confuse some applications, if they offer a username/password combination then get confused when asked for one. You can disable this behavior, so that Tor will select "No authentication" when IsolateSOCKSAuth is disabled, or when this option is set.

ExtendedErrors
Return extended error code in the SOCKS reply. So far, the possible errors are:

X'F0' Onion Service Descriptor Can Not be Found

The requested onion service descriptor can't be found on the
hashring and thus not reachable by the client. (v3 only)

X'F1' Onion Service Descriptor Is Invalid

The requested onion service descriptor can't be parsed or signature validation failed. (v3 only)

X'F2' Onion Service Introduction Failed

All introduction attempts failed either due to a combination of NACK by the intro point or time out. (v3 only)

X'F3' Onion Service Rendezvous Failed

Every rendezvous circuit has timed out and thus the client is unable to rendezvous with the service. (v3 only)

X'F4' Onion Service Missing Client Authorization

Client was able to download the requested onion service descriptor but is unable to decrypt its content because it is missing client authorization information. (v3 only)

X'F5' Onion Service Wrong Client Authorization

Client was able to download the requested onion service descriptor but is unable to decrypt its content using the client authorization information it has. This means the client access were revoked. (v3 only)

X'F6' Onion Service Invalid Address

The given .onion address is invalid. In one of these cases this error is returned: address checksum doesn't match, ed25519 public key is invalid or the encoding is invalid. (v3 only)

X'F7' Onion Service Introduction Timed Out

Similar to X'F2' code but in this case, all introduction attempts have failed due to a time out. (v3 only)

Flags are processed left to right. If flags conflict, the last flag on the line is used, and all earlier flags are ignored. No error is issued for conflicting flags.

TokenBucketRefillInterval *NUM* [**msec**|**second**]
Set the refill delay interval of Tor’s token bucket to NUM milliseconds. NUM must be between 1 and 1000, inclusive. When Tor is out of bandwidth, on a connection or globally, it will wait up to this long before it tries to use that connection again. Note that bandwidth limits are still expressed in bytes per second: this option only affects the frequency with which Tor checks to see whether previously exhausted connections may read again. Can not be changed while tor is running. (Default: 100 msec)

TrackHostExits *host,.domain,...*
For each value in the comma separated list, Tor will track recent connections to hosts that match this value and attempt to reuse the same exit node for each. If the value is prepended with a '.', it is treated as matching an entire domain. If one of the values is just a '.', it means match everything. This option is useful if you frequently connect to sites that will expire all your authentication cookies (i.e. log you out) if your IP address changes. Note that this option does have the disadvantage of making it more clear that a given history is associated with a single user. However, most people who would wish to observe this will observe it through cookies or other protocol−specific means anyhow.

TrackHostExitsExpire *NUM*

Since exit servers go up and down, it is desirable to expire the association between host and exit server after NUM seconds. The default is 1800 seconds (30 minutes).

TransPort [*address*:]*port*|**auto** [*isolation flags*]
Open this port to listen for transparent proxy connections. Set this to 0 if you don't want to allow transparent proxy connections. Set the port to "auto" to have Tor pick a port for you. This directive can be specified multiple times to bind to multiple addresses/ports. If multiple entries of this option are present in your configuration file, Tor will perform stream isolation between listeners by default. See SocksPort for an explanation of isolation flags.

TransPort requires OS support for transparent proxies, such as BSDs' pf or Linux's IPTables. If you're planning to use Tor as a transparent proxy for a network, you'll want to examine and change VirtualAddrNetwork from the default setting. (Default: 0)

TransProxyType default|**TPROXY**|**ipfw**|**pf−divert**
TransProxyType may only be enabled when there is transparent proxy listener enabled.

Set this to "TPROXY" if you wish to be able to use the TPROXY Linux module to transparently proxy connections that are configured using the TransPort option. Detailed information on how to configure the TPROXY feature can be found in the Linux kernel source tree in the file Documentation/networking/tproxy.txt.

Set this option to "ipfw" to use the FreeBSD ipfw interface.

On *BSD operating systems when using pf, set this to "pf−divert" to take advantage of divert−to rules, which do not modify the packets like rdr−to rules do. Detailed information on how to configure pf to use divert−to rules can be found in the pf.conf(5) manual page. On OpenBSD, divert−to is available to use on versions greater than or equal to OpenBSD 4.4.

Set this to "default", or leave it unconfigured, to use regular IPTables on Linux, or to use pf rdr−to rules on *BSD systems.

(Default: "default")

UpdateBridgesFromAuthority 0|**1**
When set (along with UseBridges), Tor will try to fetch bridge descriptors from the configured bridge authorities when feasible. It will fall back to a direct request if the authority responds with a 404. (Default: 0)

UseBridges 0|**1**
When set, Tor will fetch descriptors for each bridge listed in the "Bridge" config lines, and use these relays as both entry guards and directory guards. (Default: 0)

UseEntryGuards 0|**1**
If this option is set to 1, we pick a few long−term entry servers, and try to stick with them. This is desirable because constantly changing servers increases the odds that an adversary who owns some servers will observe a fraction of your paths. Entry Guards can not be used by Directory Authorities or Single Onion Services. In these cases, this option is ignored. (Default: 1)

UseGuardFraction 0|**1**|**auto**
This option specifies whether clients should use the guardfraction information found in the consensus during path selection. If it's set to *auto*, clients will do what the UseGuardFraction consensus

parameter tells them to do. (Default: auto)

GuardLifetime *N* **days**|**weeks**|**months**
If UseEntryGuards is set, minimum time to keep a guard on our guard list before picking a new one. If less than one day, we use defaults from the consensus directory. (Default: 0)

NumDirectoryGuards *NUM*
If UseEntryGuards is set to 1, we try to make sure we have at least NUM routers to use as directory guards. If this option is set to 0, use the value from the guard−n−primary−dir−guards−to−use consensus parameter, and default to 3 if the consensus parameter isn't set. (Default: 0)

NumEntryGuards *NUM*
If UseEntryGuards is set to 1, we will try to pick a total of NUM routers as long−term entries for our circuits. If NUM is 0, we try to learn the number from the guard−n−primary−guards−to−use consensus parameter, and default to 1 if the consensus parameter isn't set. (Default: 0)

NumPrimaryGuards *NUM*
If UseEntryGuards is set to 1, we will try to pick NUM routers for our primary guard list, which is the set of routers we strongly prefer when connecting to the Tor network. If NUM is 0, we try to learn the number from the guard−n−primary−guards consensus parameter, and default to 3 if the consensus parameter isn't set. (Default: 0)

UseMicrodescriptors 0|**1**|**auto**
Microdescriptors are a smaller version of the information that Tor needs in order to build its circuits. Using microdescriptors makes Tor clients download less directory information, thus saving bandwidth. Directory caches need to fetch regular descriptors and microdescriptors, so this option doesn't save any bandwidth for them. For legacy reasons, auto is accepted, but it has the same effect as 1. (Default: auto)

VirtualAddrNetworkIPv4 *IPv4Address*/*bits*

VirtualAddrNetworkIPv6 [*IPv6Address*]/*bits*
When Tor needs to assign a virtual (unused) address because of a MAPADDRESS command from the controller or the AutomapHostsOnResolve feature, Tor picks an unassigned address from this range. (Defaults: 127.192.0.0/10 and [FE80::]/10 respectively.)

When providing proxy server service to a network of computers using a tool like dns−proxy−tor, change the IPv4 network to "10.192.0.0/10" or "172.16.0.0/12" and change the IPv6 network to "[FC00::]/7". The default **VirtualAddrNetwork** address ranges on a properly configured machine will route to the loopback or link−local interface. The maximum number of bits for the network prefix is set to 104 for IPv6 and 16 for IPv4. However, a larger network (that is, one with a smaller prefix length) is preferable, since it reduces the chances for an attacker to guess the used IP. For local use, no change to the default VirtualAddrNetwork setting is needed.

CIRCUIT TIMEOUT OPTIONS

The following options are useful for configuring timeouts related to building Tor circuits and using them:

CircuitsAvailableTimeout *NUM*
Tor will attempt to keep at least one open, unused circuit available for this amount of time. This option governs how long idle circuits are kept open, as well as the amount of time Tor will keep a circuit open to each of the recently used ports. This way when the Tor client is entirely idle, it can expire all of its circuits, and then expire its TLS connections. Note that the actual timeout value is uniformly randomized from the specified value to twice that amount. (Default: 30 minutes; Max: 24 hours)

LearnCircuitBuildTimeout 0|**1**
If 0, CircuitBuildTimeout adaptive learning is disabled. (Default: 1)

CircuitBuildTimeout *NUM*
Try for at most NUM seconds when building circuits. If the circuit isn't open in that time, give up on

it. If LearnCircuitBuildTimeout is 1, this value serves as the initial value to use before a timeout is learned. If LearnCircuitBuildTimeout is 0, this value is the only value used. (Default: 60 seconds)

CircuitStreamTimeout *NUM*
If non−zero, this option overrides our internal timeout schedule for how many seconds until we detach a stream from a circuit and try a new circuit. If your network is particularly slow, you might want to set this to a number like 60. (Default: 0)

SocksTimeout *NUM*
Let a socks connection wait NUM seconds handshaking, and NUM seconds unattached waiting for an appropriate circuit, before we fail it. (Default: 2 minutes)

DORMANT MODE OPTIONS

Tor can enter dormant mode to conserve power and network bandwidth. The following options control when Tor enters and leaves dormant mode:

DormantCanceledByStartup 0|1
By default, Tor starts in active mode if it was active the last time it was shut down, and in dormant mode if it was dormant. But if this option is true, Tor treats every startup event as user activity, and Tor will never start in Dormant mode, even if it has been unused for a long time on previous runs. (Default: 0)

Note: Packagers and application developers should change the value of this option only with great caution: it has the potential to create spurious traffic on the network. This option should only be used if Tor is started by an affirmative user activity (like clicking on an application or running a command), and not if Tor is launched for some other reason (for example, by a startup process, or by an application that launches itself on every login.)

DormantClientTimeout *N* **minutes|hours|days|weeks**
If Tor spends this much time without any client activity, enter a dormant state where automatic circuits are not built, and directory information is not fetched. Does not affect servers or onion services. Must be at least 10 minutes. (Default: 24 hours)

DormantOnFirstStartup 0|1
If true, then the first time Tor starts up with a fresh DataDirectory, it starts in dormant mode, and takes no actions until the user has made a request. (This mode is recommended if installing a Tor client for a user who might not actually use it.) If false, Tor bootstraps the first time it is started, whether it sees a user request or not.

After the first time Tor starts, it begins in dormant mode if it was dormant before, and not otherwise. (Default: 0)

DormantTimeoutDisabledByIdleStreams 0|1
If true, then any open client stream (even one not reading or writing) counts as client activity for the purpose of DormantClientTimeout. If false, then only network activity counts. (Default: 1)

DormantTimeoutEnabled 0|1
If false, then no amount of time without activity is sufficient to make Tor go dormant. Setting this option to zero is only recommended for special−purpose applications that need to use the Tor binary for something other than sending or receiving Tor traffic. (Default: 1)

NODE SELECTION OPTIONS

The following options restrict the nodes that a tor client (or onion service) can use while building a circuit. These options can weaken your anonymity by making your client behavior different from other Tor clients:

EntryNodes *node,node,...*
A list of identity fingerprints and country codes of nodes to use for the first hop in your normal circuits. Normal circuits include all circuits except for direct connections to directory servers. The Bridge option overrides this option; if you have configured bridges and UseBridges is 1, the Bridges are used as your entry nodes.

This option can appear multiple times: the values from multiple lines are spliced together.

The ExcludeNodes option overrides this option: any node listed in both EntryNodes and ExcludeNodes is treated as excluded. See ExcludeNodes for more information on how to specify nodes.

ExcludeNodes *node,node,...*
A list of identity fingerprints, country codes, and address patterns of nodes to avoid when building a circuit. Country codes are 2−letter ISO3166 codes, and must be wrapped in braces; fingerprints may be preceded by a dollar sign. (Example: ExcludeNodes ABCD1234CDEF5678ABCD1234CDEF5678ABCD1234, {cc}, 255.254.0.0/8)

This option can appear multiple times: the values from multiple lines are spliced together.

By default, this option is treated as a preference that Tor is allowed to override in order to keep working. For example, if you try to connect to a hidden service, but you have excluded all of the hidden service's introduction points, Tor will connect to one of them anyway. If you do not want this behavior, set the StrictNodes option (documented below).

Note also that if you are a relay, this (and the other node selection options below) only affects your own circuits that Tor builds for you. Clients can still build circuits through you to any node. Controllers can tell Tor to build circuits through any node.

Country codes are case−insensitive. The code "{??}" refers to nodes whose country can't be identified. No country code, including {??}, works if no GeoIPFile can be loaded. See also the GeoIPExcludeUnknown option below.

ExcludeExitNodes *node,node,...*
A list of identity fingerprints, country codes, and address patterns of nodes to never use when picking an exit node−−−that is, a node that delivers traffic for you **outside** the Tor network. Note that any node listed in ExcludeNodes is automatically considered to be part of this list too. See ExcludeNodes for more information on how to specify nodes. See also the caveats on the ExitNodes option below.

This option can appear multiple times: the values from multiple lines are spliced together.

ExitNodes *node,node,...*
A list of identity fingerprints, country codes, and address patterns of nodes to use as exit node−−−that is, a node that delivers traffic for you **outside** the Tor network. See ExcludeNodes for more information on how to specify nodes.

This option can appear multiple times: the values from multiple lines are spliced together.

Note that if you list too few nodes here, or if you exclude too many exit nodes with ExcludeExitNodes, you can degrade functionality. For example, if none of the exits you list allows traffic on port 80 or 443, you won't be able to browse the web.

Note also that not every circuit is used to deliver traffic **outside** of the Tor network. It is normal to see

non−exit circuits (such as those used to connect to hidden services, those that do directory fetches, those used for relay reachability self−tests, and so on) that end at a non−exit node. To keep a node from being used entirely, see ExcludeNodes and StrictNodes.

The ExcludeNodes option overrides this option: any node listed in both ExitNodes and ExcludeNodes is treated as excluded.

The .exit address notation, if enabled via MapAddress, overrides this option.

GeoIPExcludeUnknown 0|1|auto
If this option is set to *auto*, then whenever any country code is set in ExcludeNodes or ExcludeExitNodes, all nodes with unknown country ({??} and possibly {A1}) are treated as excluded as well. If this option is set to *1*, then all unknown countries are treated as excluded in ExcludeNodes and ExcludeExitNodes. This option has no effect when a GeoIP file isn't configured or can't be found. (Default: auto)

HSLayer2Nodes *node,node,...*
A list of identity fingerprints, nicknames, country codes, and address patterns of nodes that are allowed to be used as the second hop in all client or service−side Onion Service circuits. This option mitigates attacks where the adversary runs middle nodes and induces your client or service to create many circuits, in order to discover your primary guard node. (Default: Any node in the network may be used in the second hop.)

(Example: HSLayer2Nodes ABCD1234CDEF5678ABCD1234CDEF5678ABCD1234, {cc}, 255.254.0.0/8)

This option can appear multiple times: the values from multiple lines are spliced together.

When this is set, the resulting hidden service paths will look like:

C − G − L2 − M − Rend

C − G − L2 − M − HSDir

C − G − L2 − M − Intro

S − G − L2 − M − Rend

S − G − L2 − M − HSDir

S − G − L2 − M − Intro

where C is this client, S is the service, G is the Guard node, L2 is a node from this option, and M is a random middle node. Rend, HSDir, and Intro point selection is not affected by this option.

This option may be combined with HSLayer3Nodes to create paths of the form:

C − G − L2 − L3 − Rend

C − G − L2 − L3 − M − HSDir

C − G − L2 − L3 − M − Intro

S − G − L2 − L3 − M − Rend

S − G − L2 − L3 − HSDir

S − G − L2 − L3 − Intro

ExcludeNodes have higher priority than HSLayer2Nodes, which means that nodes specified in ExcludeNodes will not be picked.

When either this option or HSLayer3Nodes are set, the /16 subnet and node family restrictions are removed for hidden service circuits. Additionally, we allow the guard node to be present as the Rend, HSDir, and IP node, and as the hop before it. This is done to prevent the adversary from inferring information about our guard, layer2, and layer3 node choices at later points in the path.

This option is meant to be managed by a Tor controller such as https://github.com/mikeperry−tor/vanguards that selects and updates this set of nodes for you. Hence it does not do load balancing if fewer than 20 nodes are selected, and if no nodes in HSLayer2Nodes are currently available for use, Tor will not work. Please use extreme care if you are setting this option manually.

HSLayer3Nodes *node,node,...*
A list of identity fingerprints, nicknames, country codes, and address patterns of nodes that are allowed to be used as the third hop in all client and service−side Onion Service circuits. This option mitigates attacks where the adversary runs middle nodes and induces your client or service to create many circuits, in order to discover your primary or Layer2 guard nodes. (Default: Any node in the network may be used in the third hop.)

(Example: HSLayer3Nodes ABCD1234CDEF5678ABCD1234CDEF5678ABCD1234, {cc}, 255.254.0.0/8)

This option can appear multiple times: the values from multiple lines are spliced together.

When this is set by itself, the resulting hidden service paths will look like:

C − G − M − L3 − Rend

C − G − M − L3 − M − HSDir

C − G − M − L3 − M − Intro

S − G − M − L3 − M − Rend

S − G − M − L3 − HSDir

S − G − M − L3 − Intro

where C is this client, S is the service, G is the Guard node, L2 is a node from this option, and M is a random middle node. Rend, HSDir, and Intro point selection is not affected by this option.

While it is possible to use this option by itself, it should be combined with HSLayer2Nodes to create

paths of the form:

C − G − L2 − L3 − Rend

C − G − L2 − L3 − M − HSDir

C − G − L2 − L3 − M − Intro

S − G − L2 − L3 − M − Rend

S − G − L2 − L3 − HSDir

S − G − L2 − L3 − Intro

ExcludeNodes have higher priority than HSLayer3Nodes, which means that nodes specified in ExcludeNodes will not be picked.

When either this option or HSLayer2Nodes are set, the /16 subnet and node family restrictions are removed for hidden service circuits. Additionally, we allow the guard node to be present as the Rend, HSDir, and IP node, and as the hop before it. This is done to prevent the adversary from inferring information about our guard, layer2, and layer3 node choices at later points in the path.

This option is meant to be managed by a Tor controller such as https://github.com/mikeperry−tor/vanguards that selects and updates this set of nodes for you. Hence it does not do load balancing if fewer than 20 nodes are selected, and if no nodes in HSLayer3Nodes are currently available for use, Tor will not work. Please use extreme care if you are setting this option manually.

MiddleNodes *node,node,...*
A list of identity fingerprints and country codes of nodes to use for "middle" hops in your normal circuits. Normal circuits include all circuits except for direct connections to directory servers. Middle hops are all hops other than exit and entry.

This option can appear multiple times: the values from multiple lines are spliced together.

This is an **experimental** feature that is meant to be used by researchers and developers to test new features in the Tor network safely. Using it without care will strongly influence your anonymity. Other tor features may not work with MiddleNodes. This feature might get removed in the future.

The HSLayer2Node and HSLayer3Node options override this option for onion service circuits, if they are set. The vanguards addon will read this option, and if set, it will set HSLayer2Nodes and HSLayer3Nodes to nodes from this set.

The ExcludeNodes option overrides this option: any node listed in both MiddleNodes and ExcludeNodes is treated as excluded. See the <<ExcludeNodes,ExcludeNodes>> for more information on how to specify nodes.

NodeFamily *node,node,...*
The Tor servers, defined by their identity fingerprints, constitute a "family" of similar or co−administered servers, so never use any two of them in the same circuit. Defining a NodeFamily is only needed when a server doesn’t list the family itself (with MyFamily). This option can be used multiple times; each instance defines a separate family. In addition to nodes, you can also list IP address and ranges and country codes in {curly braces}. See ExcludeNodes for more information on

how to specify nodes.

StrictNodes 0|1
If StrictNodes is set to 1, Tor will treat solely the ExcludeNodes option as a requirement to follow for all the circuits you generate, even if doing so will break functionality for you (StrictNodes does not apply to ExcludeExitNodes, ExitNodes, MiddleNodes, or MapAddress). If StrictNodes is set to 0, Tor will still try to avoid nodes in the ExcludeNodes list, but it will err on the side of avoiding unexpected errors. Specifically, StrictNodes 0 tells Tor that it is okay to use an excluded node when it is **necessary** to perform relay reachability self−tests, connect to a hidden service, provide a hidden service to a client, fulfill a .exit request, upload directory information, or download directory information. (Default: 0)

SERVER OPTIONS

The following options are useful only for servers (that is, if ORPort is non−zero):

AccountingMax *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
Limits the max number of bytes sent and received within a set time period using a given calculation rule (see AccountingStart and AccountingRule). Useful if you need to stay under a specific bandwidth. By default, the number used for calculation is the max of either the bytes sent or received. For example, with AccountingMax set to 1 TByte, a server could send 900 GBytes and receive 800 GBytes and continue running. It will only hibernate once one of the two reaches 1 TByte. This can be changed to use the sum of the both bytes received and sent by setting the AccountingRule option to "sum" (total bandwidth in/out). When the number of bytes remaining gets low, Tor will stop accepting new connections and circuits. When the number of bytes is exhausted, Tor will hibernate until some time in the next accounting period. To prevent all servers from waking at the same time, Tor will also wait until a random point in each period before waking up. If you have bandwidth cost issues, enabling hibernation is preferable to setting a low bandwidth, since it provides users with a collection of fast servers that are up some of the time, which is more useful than a set of slow servers that are always "available".

Note that (as also described in the Bandwidth section) Tor uses powers of two, not powers of ten: 1 GByte is 1024*1024*1024, not one billion. Be careful: some internet service providers might count GBytes differently.

AccountingRule sum|max|in|out
How we determine when our AccountingMax has been reached (when we should hibernate) during a time interval. Set to "max" to calculate using the higher of either the sent or received bytes (this is the default functionality). Set to "sum" to calculate using the sent plus received bytes. Set to "in" to calculate using only the received bytes. Set to "out" to calculate using only the sent bytes. (Default: max)

AccountingStart day|week|month [*day*] *HH:MM*
Specify how long accounting periods last. If **month** is given, each accounting period runs from the time *HH:MM* on the *dayth* day of one month to the same day and time of the next. The relay will go at full speed, use all the quota you specify, then hibernate for the rest of the period. (The day must be between 1 and 28.) If **week** is given, each accounting period runs from the time *HH:MM* of the *dayth* day of one week to the same day and time of the next week, with Monday as day 1 and Sunday as day 7. If **day** is given, each accounting period runs from the time *HH:MM* each day to the same time on the next day. All times are local, and given in 24−hour time. (Default: "month 1 0:00")

Address *address*
The address of this server, or a fully qualified domain name of this server that resolves to an address. You can leave this unset, and Tor will try to guess your address. If a domain name is provided, Tor will attempt to resolve it and use the underlying IPv4/IPv6 address as its publish address (taking precedence over the ORPort configuration). The publish address is the one used to tell clients and other servers where to find your Tor server; it doesn’t affect the address that your server binds to. To bind to a different address, use the ORPort and OutboundBindAddress options.

AddressDisableIPv6 0|1
By default, Tor will attempt to find the IPv6 of the relay if there is no IPv4Only ORPort. If set, this option disables IPv6 auto discovery. This disables IPv6 address resolution, IPv6 ORPorts, and IPv6 reachability checks. Also, the relay won't publish an IPv6 ORPort in its descriptor. (Default: 0)

AssumeReachable 0|1
This option is used when bootstrapping a new Tor network. If set to 1, don't do self−reachability testing; just upload your server descriptor immediately. (Default: 0)

AssumeReachableIPv6 0|1|auto
Like **AssumeReachable**, but affects only the relay's own IPv6 ORPort. If this value is set to "auto", then Tor will look at **AssumeReachable** instead. (Default: auto)

BridgeRelay 0|1
Sets the relay to act as a "bridge" with respect to relaying connections from bridge users to the Tor network. It mainly causes Tor to publish a server descriptor to the bridge database, rather than to the public directory authorities.

Note: make sure that no MyFamily lines are present in your torrc when relay is configured in bridge mode.

BridgeDistribution *string*
If set along with BridgeRelay, Tor will include a new line in its bridge descriptor which indicates to the BridgeDB service how it would like its bridge address to be given out. Set it to "none" if you want BridgeDB to avoid distributing your bridge address, or "any" to let BridgeDB decide. See https://bridges.torproject.org/info for a more up−to−date list of options. (Default: any)

ContactInfo *email_address*
Administrative contact information for this relay or bridge. This line can be used to contact you if your relay or bridge is misconfigured or something else goes wrong. Note that we archive and publish all descriptors containing these lines and that Google indexes them, so spammers might also collect them. You may want to obscure the fact that it's an email address and/or generate a new address for this purpose.

ContactInfo **must** be set to a working address if you run more than one relay or bridge. (Really, everybody running a relay or bridge should set it.)

DisableOOSCheck 0|1
This option disables the code that closes connections when Tor notices that it is running low on sockets. Right now, it is on by default, since the existing out−of−sockets mechanism tends to kill OR connections more than it should. (Default: 1)

ExitPolicy *policy,policy,...*
Set an exit policy for this server. Each policy is of the form "**accept[6]|reject[6]** *ADDR*[/*MASK*][:*PORT*]". If /*MASK* is omitted then this policy just applies to the host given. Instead of giving a host or network you can also use "*" to denote the universe (0.0.0.0/0 and ::/0), or *4 to denote all IPv4 addresses, and *6 to denote all IPv6 addresses. *PORT* can be a single port number, an interval of ports "*FROM_PORT−TO_PORT*", or "*". If *PORT* is omitted, that means "*".

For example, "accept 18.7.22.69:*,reject 18.0.0.0/8:*,accept *:*" would reject any IPv4 traffic destined for MIT except for web.mit.edu, and accept any other IPv4 or IPv6 traffic.

Tor also allows IPv6 exit policy entries. For instance, "reject6 [FC00::]/7:*" rejects all destinations that share 7 most significant bit prefix with address FC00::. Respectively, "accept6 [C000::]/3:*" accepts all destinations that share 3 most significant bit prefix with address C000::.

accept6 and reject6 only produce IPv6 exit policy entries. Using an IPv4 address with accept6 or reject6 is ignored and generates a warning. accept/reject allows either IPv4 or IPv6 addresses. Use *4 as an IPv4 wildcard address, and *6 as an IPv6 wildcard address. accept/reject * expands to matching IPv4 and IPv6 wildcard address rules.

To specify all IPv4 and IPv6 internal and link−local networks (including 0.0.0.0/8, 169.254.0.0/16, 127.0.0.0/8, 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12, [::]/8, [FC00::]/7, [FE80::]/10, [FEC0::]/10, [FF00::]/8, and [::]/127), you can use the "private" alias instead of an address. ("private" always produces rules for IPv4 and IPv6 addresses, even when used with accept6/reject6.)

Private addresses are rejected by default (at the beginning of your exit policy), along with any configured primary public IPv4 and IPv6 addresses. These private addresses are rejected unless you set the ExitPolicyRejectPrivate config option to 0. For example, once you've done that, you could allow HTTP to 127.0.0.1 and block all other connections to internal networks with "accept 127.0.0.1:80,reject private:*", though that may also allow connections to your own computer that are addressed to its public (external) IP address. See RFC 1918 and RFC 3330 for more details about internal and reserved IP address space. See ExitPolicyRejectLocalInterfaces if you want to block every address on the relay, even those that aren't advertised in the descriptor.

This directive can be specified multiple times so you don't have to put it all on one line.

Policies are considered first to last, and the first match wins. If you want to allow the same ports on IPv4 and IPv6, write your rules using accept/reject *. If you want to allow different ports on IPv4 and IPv6, write your IPv6 rules using accept6/reject6 *6, and your IPv4 rules using accept/reject *4. If you want to _replace_ the default exit policy, end your exit policy with either a reject *:* or an accept *:*. Otherwise, you're _augmenting_ (prepending to) the default exit policy.

If you want to use a reduced exit policy rather than the default exit policy, set "ReducedExitPolicy 1". If you want to *replace* the default exit policy with your custom exit policy, end your exit policy with either a reject **:** or an accept **:**. Otherwise, you're *augmenting* (prepending to) the default or reduced exit policy.

The default exit policy is:

reject *:25
reject *:119
reject *:135−139
reject *:445
reject *:563
reject *:1214
reject *:4661−4666
reject *:6346−6429
reject *:6699
reject *:6881−6999
accept *:*

Since the default exit policy uses accept/reject *, it applies to both IPv4 and IPv6 addresses.

ExitPolicyRejectLocalInterfaces 0|1
Reject all IPv4 and IPv6 addresses that the relay knows about, at the beginning of your exit policy.

This includes any OutboundBindAddress, the bind addresses of any port options, such as ControlPort or DNSPort, and any public IPv4 and IPv6 addresses on any interface on the relay. (If IPv6Exit is not set, all IPv6 addresses will be rejected anyway.) See above entry on ExitPolicy. This option is off by default, because it lists all public relay IP addresses in the ExitPolicy, even those relay operators might prefer not to disclose. (Default: 0)

ExitPolicyRejectPrivate 0|1
Reject all private (local) networks, along with the relay's advertised public IPv4 and IPv6 addresses, at the beginning of your exit policy. See above entry on ExitPolicy. (Default: 1)

ExitRelay 0|1|auto
Tells Tor whether to run as an exit relay. If Tor is running as a non−bridge server, and ExitRelay is set to 1, then Tor allows traffic to exit according to the ExitPolicy option, the ReducedExitPolicy option, or the default ExitPolicy (if no other exit policy option is specified).

If ExitRelay is set to 0, no traffic is allowed to exit, and the ExitPolicy, ReducedExitPolicy, and IPv6Exit options are ignored.

If ExitRelay is set to "auto", then Tor checks the ExitPolicy, ReducedExitPolicy, and IPv6Exit options. If at least one of these options is set, Tor behaves as if ExitRelay were set to 1. If none of these exit policy options are set, Tor behaves as if ExitRelay were set to 0. (Default: auto)

ExtendAllowPrivateAddresses 0|1
When this option is enabled, Tor will connect to relays on localhost, RFC1918 addresses, and so on. In particular, Tor will make direct OR connections, and Tor routers allow EXTEND requests, to these private addresses. (Tor will always allow connections to bridges, proxies, and pluggable transports configured on private addresses.) Enabling this option can create security issues; you should probably leave it off. (Default: 0)

GeoIPFile *filename*
A filename containing IPv4 GeoIP data, for use with by−country statistics.

GeoIPv6File *filename*
A filename containing IPv6 GeoIP data, for use with by−country statistics.

HeartbeatPeriod *N* **minutes|hours|days|weeks**
Log a heartbeat message every **HeartbeatPeriod** seconds. This is a log level *notice* message, designed to let you know your Tor server is still alive and doing useful things. Settings this to 0 will disable the heartbeat. Otherwise, it must be at least 30 minutes. (Default: 6 hours)

IPv6Exit 0|1
If set, and we are an exit node, allow clients to use us for IPv6 traffic. When this option is set and ExitRelay is auto, we act as if ExitRelay is 1. (Default: 0)

KeyDirectory *DIR*
Store secret keys in DIR. Can not be changed while tor is running. (Default: the "keys" subdirectory of DataDirectory.)

KeyDirectoryGroupReadable 0|1|auto
If this option is set to 0, don't allow the filesystem group to read the KeyDirectory. If the option is set to 1, make the KeyDirectory readable by the default GID. If the option is "auto", then we use the setting for DataDirectoryGroupReadable when the KeyDirectory is the same as the DataDirectory, and 0 otherwise. (Default: auto)

MainloopStats 0|1
Log main loop statistics every **HeartbeatPeriod** seconds. This is a log level *notice* message designed to help developers instrumenting Tor's main event loop. (Default: 0)

MaxMemInQueues *N* **bytes|KBytes|MBytes|GBytes**

This option configures a threshold above which Tor will assume that it needs to stop queueing or buffering data because it’s about to run out of memory. If it hits this threshold, it will begin killing circuits until it has recovered at least 10% of this memory. Do not set this option too low, or your relay may be unreliable under load. This option only affects some queues, so the actual process size will be larger than this. If this option is set to 0, Tor will try to pick a reasonable default based on your system’s physical memory. (Default: 0)

MaxOnionQueueDelay *NUM* [**msec**|**second**]
If we have more onionskins queued for processing than we can process in this amount of time, reject new ones. (Default: 1750 msec)

MyFamily *fingerprint*,*fingerprint*,...
Declare that this Tor relay is controlled or administered by a group or organization identical or similar to that of the other relays, defined by their (possibly $−prefixed) identity fingerprints. This option can be repeated many times, for convenience in defining large families: all fingerprints in all MyFamily lines are merged into one list. When two relays both declare that they are in the same 'family', Tor clients will not use them in the same circuit. (Each relay only needs to list the other servers in its family; it doesn’t need to list itself, but it won’t hurt if it does.) Do not list any bridge relay as it would compromise its concealment.

If you run more than one relay, the MyFamily option on each relay **must** list all other relays, as described above.

Note: do not use MyFamily when configuring your Tor instance as a bridge.

Nickname *name*
Set the server’s nickname to 'name'. Nicknames must be between 1 and 19 characters inclusive, and must contain only the characters [a−zA−Z0−9]. If not set, **Unnamed** will be used. Relays can always be uniquely identified by their identity fingerprints.

NumCPUs *num*
How many processes to use at once for decrypting onionskins and other parallelizable operations. If this is set to 0, Tor will try to detect how many CPUs you have, defaulting to 1 if it can’t tell. (Default: 0)

OfflineMasterKey 0|**1**
If non−zero, the Tor relay will never generate or load its master secret key. Instead, you’ll have to use "tor −−keygen" to manage the permanent ed25519 master identity key, as well as the corresponding temporary signing keys and certificates. (Default: 0)

ORPort [*address***:**]*PORT*|**auto** [*flags*]
Advertise this port to listen for connections from Tor clients and servers. This option is required to be a Tor server. Set it to "auto" to have Tor pick a port for you. Set it to 0 to not run an ORPort at all. This option can occur more than once. (Default: 0)

Tor recognizes these flags on each ORPort:

NoAdvertise
By default, we bind to a port and tell our users about it. If NoAdvertise is specified, we don’t advertise, but listen anyway. This can be useful if the port everybody will be connecting to (for example, one that’s opened on our firewall) is somewhere else.

NoListen
By default, we bind to a port and tell our users about it. If NoListen is specified, we don’t bind, but advertise anyway. This can be useful if something else (for example, a firewall’s port forwarding configuration) is causing connections to reach us.

IPv4Only
If the address is absent, or resolves to both an IPv4 and an IPv6 address, only listen to the IPv4 address.

IPv6Only
If the address is absent, or resolves to both an IPv4 and an IPv6 address, only listen to the IPv6 address.

For obvious reasons, NoAdvertise and NoListen are mutually exclusive, and IPv4Only and IPv6Only are mutually exclusive.

PublishServerDescriptor 0|1|v3|bridge,...
This option specifies which descriptors Tor will publish when acting as a relay. You can choose multiple arguments, separated by commas.

If this option is set to 0, Tor will not publish its descriptors to any directories. (This is useful if you're testing out your server, or if you're using a Tor controller that handles directory publishing for you.) Otherwise, Tor will publish its descriptors of all type(s) specified. The default is "1", which means "if running as a relay or bridge, publish descriptors to the appropriate authorities". Other possibilities are "v3", meaning "publish as if you're a relay", and "bridge", meaning "publish as if you're a bridge".

ReducedExitPolicy 0|1
If set, use a reduced exit policy rather than the default one.

The reduced exit policy is an alternative to the default exit policy. It allows as many Internet services as possible while still blocking the majority of TCP ports. Currently, the policy allows approximately 65 ports. This reduces the odds that your node will be used for peer−to−peer applications.

The reduced exit policy is:

accept *:20−21
accept *:22
accept *:23
accept *:43
accept *:53
accept *:79
accept *:80−81
accept *:88
accept *:110
accept *:143
accept *:194
accept *:220
accept *:389
accept *:443
accept *:464
accept *:465
accept *:531
accept *:543−544
accept *:554
accept *:563
accept *:587
accept *:636
accept *:706
accept *:749

accept *:873
accept *:902−904
accept *:981
accept *:989−990
accept *:991
accept *:992
accept *:993
accept *:994
accept *:995
accept *:1194
accept *:1220
accept *:1293
accept *:1500
accept *:1533
accept *:1677
accept *:1723
accept *:1755
accept *:1863
accept *:2082
accept *:2083
accept *:2086−2087
accept *:2095−2096
accept *:2102−2104
accept *:3128
accept *:3389
accept *:3690
accept *:4321
accept *:4643
accept *:5050
accept *:5190
accept *:5222−5223
accept *:5228
accept *:5900
accept *:6660−6669
accept *:6679
accept *:6697
accept *:8000
accept *:8008
accept *:8074
accept *:8080
accept *:8082
accept *:8087−8088
accept *:8232−8233
accept *:8332−8333
accept *:8443
accept *:8888
accept *:9418
accept *:9999
accept *:10000
accept *:11371
accept *:19294
accept *:19638
accept *:50002
accept *:64738

reject *:*

(Default: 0)

RefuseUnknownExits 0|1|auto
Prevent nodes that don’t appear in the consensus from exiting using this relay. If the option is 1, we always block exit attempts from such nodes; if it’s 0, we never do, and if the option is "auto", then we do whatever the authorities suggest in the consensus (and block if the consensus is quiet on the issue). (Default: auto)

ServerDNSAllowBrokenConfig 0|1
If this option is false, Tor exits immediately if there are problems parsing the system DNS configuration or connecting to nameservers. Otherwise, Tor continues to periodically retry the system nameservers until it eventually succeeds. (Default: 1)

ServerDNSAllowNonRFC953Hostnames 0|1
When this option is disabled, Tor does not try to resolve hostnames containing illegal characters (like @ and :) rather than sending them to an exit node to be resolved. This helps trap accidental attempts to resolve URLs and so on. This option only affects name lookups that your server does on behalf of clients. (Default: 0)

ServerDNSDetectHijacking 0|1
When this option is set to 1, we will test periodically to determine whether our local nameservers have been configured to hijack failing DNS requests (usually to an advertising site). If they are, we will attempt to correct this. This option only affects name lookups that your server does on behalf of clients. (Default: 1)

ServerDNSRandomizeCase 0|1
When this option is set, Tor sets the case of each character randomly in outgoing DNS requests, and makes sure that the case matches in DNS replies. This so−called "0x20 hack" helps resist some types of DNS poisoning attack. For more information, see "Increased DNS Forgery Resistance through 0x20−Bit Encoding". This option only affects name lookups that your server does on behalf of clients. (Default: 1)

ServerDNSResolvConfFile *filename*
Overrides the default DNS configuration with the configuration in *filename*. The file format is the same as the standard Unix "**resolv.conf**" file (7). This option, like all other ServerDNS options, only affects name lookups that your server does on behalf of clients. (Defaults to use the system DNS configuration or a localhost DNS service in case no nameservers are found in a given configuration.)

ServerDNSSearchDomains 0|1
If set to 1, then we will search for addresses in the local search domain. For example, if this system is configured to believe it is in "example.com", and a client tries to connect to "www", the client will be connected to "www.example.com". This option only affects name lookups that your server does on behalf of clients. (Default: 0)

ServerDNSTestAddresses *hostname,hostname,...*
When we’re detecting DNS hijacking, make sure that these *valid* addresses aren’t getting redirected. If they are, then our DNS is completely useless, and we’ll reset our exit policy to "reject *:*". This option only affects name lookups that your server does on behalf of clients. (Default: "www.google.com, www.mit.edu, www.yahoo.com, www.slashdot.org")

ServerTransportListenAddr *transport IP*:*PORT*
When this option is set, Tor will suggest *IP*:*PORT* as the listening address of any pluggable transport proxy that tries to launch *transport*. (IPv4 addresses should written as−is; IPv6 addresses should be wrapped in square brackets.) (Default: none)

ServerTransportOptions *transport k=v k=v ...*
When this option is set, Tor will pass the *k=v* parameters to any pluggable transport proxy that tries to launch *transport*.

(Example: ServerTransportOptions obfs45 shared−secret=bridgepasswd cache=/var/lib/tor/cache) (Default: none)

ServerTransportPlugin *transport* exec *path−to−binary* [options]
The Tor relay launches the pluggable transport proxy in *path−to−binary* using *options* as its command−line options, and expects to receive proxied client traffic from it. (Default: none)

ShutdownWaitLength *NUM*
When we get a SIGINT and we're a server, we begin shutting down: we close listeners and start refusing new circuits. After **NUM** seconds, we exit. If we get a second SIGINT, we exit immediately. (Default: 30 seconds)

SigningKeyLifetime *N* **days|weeks|months**
For how long should each Ed25519 signing key be valid? Tor uses a permanent master identity key that can be kept offline, and periodically generates new "signing" keys that it uses online. This option configures their lifetime. (Default: 30 days)

SSLKeyLifetime *N* **minutes|hours|days|weeks**
When creating a link certificate for our outermost SSL handshake, set its lifetime to this amount of time. If set to 0, Tor will choose some reasonable random defaults. (Default: 0)

STATISTICS OPTIONS

Relays publish most statistics in a document called the extra−info document. The following options affect the different types of statistics that Tor relays collect and publish:

BridgeRecordUsageByCountry 0|1
When this option is enabled and BridgeRelay is also enabled, and we have GeoIP data, Tor keeps a per−country count of how many client addresses have contacted it so that it can help the bridge authority guess which countries have blocked access to it. If ExtraInfoStatistics is enabled, it will be published as part of the extra−info document. (Default: 1)

CellStatistics 0|1
Relays only. When this option is enabled, Tor collects statistics about cell processing (i.e. mean time a cell is spending in a queue, mean number of cells in a queue and mean number of processed cells per circuit) and writes them into disk every 24 hours. Onion router operators may use the statistics for performance monitoring. If ExtraInfoStatistics is enabled, it will published as part of the extra−info document. (Default: 0)

ConnDirectionStatistics 0|1
Relays only. When this option is enabled, Tor writes statistics on the amounts of traffic it passes between itself and other relays to disk every 24 hours. Enables relay operators to monitor how much their relay is being used as middle node in the circuit. If ExtraInfoStatistics is enabled, it will be published as part of the extra−info document. (Default: 0)

DirReqStatistics 0|1
Relays and bridges only. When this option is enabled, a Tor directory writes statistics on the number and response time of network status requests to disk every 24 hours. Enables relay and bridge operators to monitor how much their server is being used by clients to learn about Tor network. If ExtraInfoStatistics is enabled, it will published as part of the extra−info document. (Default: 1)

EntryStatistics 0|1
Relays only. When this option is enabled, Tor writes statistics on the number of directly connecting clients to disk every 24 hours. Enables relay operators to monitor how much inbound traffic that originates from Tor clients passes through their server to go further down the Tor network. If ExtraInfoStatistics is enabled, it will be published as part of the extra−info document. (Default: 0)

ExitPortStatistics 0|1
Exit relays only. When this option is enabled, Tor writes statistics on the number of relayed bytes and opened stream per exit port to disk every 24 hours. Enables exit relay operators to measure and monitor amounts of traffic that leaves Tor network through their exit node. If ExtraInfoStatistics is enabled, it will be published as part of the extra−info document. (Default: 0)

ExtraInfoStatistics 0|1
When this option is enabled, Tor includes previously gathered statistics in its extra−info documents that it uploads to the directory authorities. Disabling this option also removes bandwidth usage statistics, and GeoIPFile and GeoIPv6File hashes from the extra−info file. Bridge ServerTransportPlugin lines are always included in the extra−info file, because they are required by BridgeDB. (Default: 1)

HiddenServiceStatistics 0|1
Relays and bridges only. When this option is enabled, a Tor relay writes obfuscated statistics on its role as hidden−service directory, introduction point, or rendezvous point to disk every 24 hours. If ExtraInfoStatistics is enabled, it will be published as part of the extra−info document. (Default: 1)

OverloadStatistics 0|1*
Relays and bridges only. When this option is enabled, a Tor relay will write an overload general line in the server descriptor if the relay is considered overloaded. (Default: 1)

A relay is considered overloaded if at least one of these conditions is met:

- Onionskins are starting to be dropped.
- The OOM was invoked.
- (Exit only) DNS timeout occurs X% of the time over Y seconds (values controlled by consensus parameters, see param−spec.txt).

  If ExtraInfoStatistics is enabled, it can also put two more specific overload lines in the extra−info document if at least one of these conditions is met:

- TCP Port exhaustion.
- Connection rate limits have been reached (read and write side).

PaddingStatistics 0|1
Relays and bridges only. When this option is enabled, Tor collects statistics for padding cells sent and received by this relay, in addition to total cell counts. These statistics are rounded, and omitted if traffic is low. This information is important for load balancing decisions related to padding. If ExtraInfoStatistics is enabled, it will be published as a part of the extra−info document. (Default: 1)

DIRECTORY SERVER OPTIONS

The following options are useful only for directory servers. (Relays with enough bandwidth automatically become directory servers; see DirCache for details.)

DirCache 0|1
When this option is set, Tor caches all current directory documents except extra info documents, and accepts client requests for them. If **DownloadExtraInfo** is set, cached extra info documents are also cached. Setting **DirPort** is not required for **DirCache**, because clients connect via the ORPort by default. Setting either DirPort or BridgeRelay and setting DirCache to 0 is not supported. (Default: 1)

DirPolicy *policy*,*policy*,...
Set an entrance policy for this server, to limit who can connect to the directory ports. The policies have the same form as exit policies above, except that port specifiers are ignored. Any address not matched by some entry in the policy is accepted.

DirPort [*address*:]*PORT*|**auto** [*flags*]
If this option is nonzero, advertise the directory service on this port. Set it to "auto" to have Tor pick a port for you. This option can occur more than once, but only one advertised DirPort is supported: all but one DirPort must have the **NoAdvertise** flag set. (Default: 0)

The same flags are supported here as are supported by ORPort. This port can only be IPv4.

As of Tor 0.4.6.1−alpha, non−authoritative relays (see AuthoritativeDirectory) will not publish the

DirPort but will still listen on it. Clients don't use the DirPorts on relays, so it is safe for you to remove the DirPort from your torrc configuration.

DirPortFrontPage *FILENAME*
When this option is set, it takes an HTML file and publishes it as "/" on the DirPort. Now relay operators can provide a disclaimer without needing to set up a separate webserver. There's a sample disclaimer in contrib/operator−tools/tor−exit−notice.html.

MaxConsensusAgeForDiffs *N* **minutes**|**hours**|**days**|**weeks**
When this option is nonzero, Tor caches will not try to generate consensus diffs for any consensus older than this amount of time. If this option is set to zero, Tor will pick a reasonable default from the current networkstatus document. You should not set this option unless your cache is severely low on disk space or CPU. If you need to set it, keeping it above 3 or 4 hours will help clients much more than setting it to zero. (Default: 0)

DENIAL OF SERVICE MITIGATION OPTIONS

Tor has three built−in mitigation options that can be individually enabled/disabled and fine−tuned, but by default Tor directory authorities will define reasonable values for relays and no explicit configuration is required to make use of these protections. The mitigations take place at relays, and are as follows:

1. If a single client address makes too many concurrent connections (this is configurable via DoSConnectionMaxConcurrentCount), hang up on further connections.
2. If a single client IP address (v4 or v6) makes circuits too quickly (default values are more than 3 per second, with an allowed burst of 90, see DoSCircuitCreationRate and DoSCircuitCreationBurst) while also having too many connections open (default is 3, see DoSCircuitCreationMinConnections), tor will refuse any new circuit (CREATE cells) for the next while (random value between 1 and 2 hours).
3. If a client asks to establish a rendezvous point to you directly (ex: Tor2Web client), ignore the request.

These defenses can be manually controlled by torrc options, but relays will also take guidance from consensus parameters using these same names, so there's no need to configure anything manually. In doubt, do not change those values.

The values set by the consensus, if any, can be found here:
https://consensus−health.torproject.org/#consensusparams

If any of the DoS mitigations are enabled, a heartbeat message will appear in your log at NOTICE level which looks like:

DoS mitigation since startup: 429042 circuits rejected, 17 marked addresses.
2238 connections closed. 8052 single hop clients refused.

The following options are useful only for a public relay. They control the Denial of Service mitigation subsystem described above.

DoSCircuitCreationEnabled 0|**1**|**auto**
Enable circuit creation DoS mitigation. If set to 1 (enabled), tor will cache client IPs along with statistics in order to detect circuit DoS attacks. If an address is positively identified, tor will activate defenses against the address. See DoSCircuitCreationDefenseType option for more details. This is a client to relay detection only. "auto" means use the consensus parameter. If not defined in the consensus, the value is 0. (Default: auto)

DoSCircuitCreationBurst *NUM*
The allowed circuit creation burst per client IP address. If the circuit rate and the burst are reached, a client is marked as executing a circuit creation DoS. "0" means use the consensus parameter. If not defined in the consensus, the value is 90. (Default: 0)

DoSCircuitCreationDefenseTimePeriod *N* **seconds**|**minutes**|**hours**
The base time period in seconds that the DoS defense is activated for. The actual value is selected randomly for each activation from N+1 to 3/2 * N. "0" means use the consensus parameter. If not defined in the consensus, the value is 3600 seconds (1 hour). (Default: 0)

DoSCircuitCreationDefenseType *NUM*
This is the type of defense applied to a detected client address. The possible values are:

1: No defense.

2: Refuse circuit creation for the DoSCircuitCreationDefenseTimePeriod period of time.

"0" means use the consensus parameter. If not defined in the consensus, the value is 2. (Default: 0)

DoSCircuitCreationMinConnections *NUM*
Minimum threshold of concurrent connections before a client address can be flagged as executing a circuit creation DoS. In other words, once a client address reaches the circuit rate and has a minimum of NUM concurrent connections, a detection is positive. "0" means use the consensus parameter. If not defined in the consensus, the value is 3. (Default: 0)

DoSCircuitCreationRate *NUM*
The allowed circuit creation rate per second applied per client IP address. If this option is 0, it obeys a consensus parameter. If not defined in the consensus, the value is 3. (Default: 0)

DoSConnectionEnabled 0|**1**|**auto**
Enable the connection DoS mitigation. If set to 1 (enabled), for client address only, this allows tor to mitigate against large number of concurrent connections made by a single IP address. "auto" means use the consensus parameter. If not defined in the consensus, the value is 0. (Default: auto)

DoSConnectionDefenseType *NUM*
This is the type of defense applied to a detected client address for the connection mitigation. The possible values are:

1: No defense.

2: Immediately close new connections.

"0" means use the consensus parameter. If not defined in the consensus, the value is 2. (Default: 0)

DoSConnectionMaxConcurrentCount *NUM*
The maximum threshold of concurrent connection from a client IP address. Above this limit, a defense selected by DoSConnectionDefenseType is applied. "0" means use the consensus parameter. If not defined in the consensus, the value is 100. (Default: 0)

DoSConnectionConnectRate *NUM*
The allowed rate of client connection from a single address per second. Coupled with the burst (see below), if the limit is reached, the address is marked and a defense is applied (DoSConnectionDefenseType) for a period of time defined by DoSConnectionConnectDefenseTimePeriod. If not defined or set to 0, it is controlled by a consensus parameter. (Default: 0)

DoSConnectionConnectBurst *NUM*
The allowed burst of client connection from a single address per second. See the DoSConnectionConnectRate for more details on this detection. If not defined or set to 0, it is controlled by a consensus parameter. (Default: 0)

DoSConnectionConnectDefenseTimePeriod *N* **seconds**|**minutes**|**hours**
The base time period in seconds that the client connection defense is activated for. The actual value is selected randomly for each activation from N+1 to 3/2 * N. If not defined or set to 0, it is controlled by a consensus parameter. (Default: 24 hours)

DoSRefuseSingleHopClientRendezvous 0|1|auto
Refuse establishment of rendezvous points for single hop clients. In other words, if a client directly connects to the relay and sends an ESTABLISH_RENDEZVOUS cell, it is silently dropped. "auto" means use the consensus parameter. If not defined in the consensus, the value is 0. (Default: auto)

DIRECTORY AUTHORITY SERVER OPTIONS

The following options enable operation as a directory authority, and control how Tor behaves as a directory authority. You should not need to adjust any of them if you're running a regular relay or exit server on the public Tor network.

AuthoritativeDirectory 0|1
When this option is set to 1, Tor operates as an authoritative directory server. Instead of caching the directory, it generates its own list of good servers, signs it, and sends that to the clients. Unless the clients already have you listed as a trusted directory, you probably do not want to set this option.

BridgeAuthoritativeDir 0|1
When this option is set in addition to **AuthoritativeDirectory**, Tor accepts and serves server descriptors, but it caches and serves the main networkstatus documents rather than generating its own. (Default: 0)

V3AuthoritativeDirectory 0|1
When this option is set in addition to **AuthoritativeDirectory**, Tor generates version 3 network statuses and serves descriptors, etc as described in dir−spec.txt file of torspec (for Tor clients and servers running at least 0.2.0.x).

AuthDirBadExit *AddressPattern...*
Authoritative directories only. A set of address patterns for servers that will be listed as bad exits in any network status document this authority publishes, if **AuthDirListBadExits** is set.

(The address pattern syntax here and in the options below is the same as for exit policies, except that you don't need to say "accept" or "reject", and ports are not needed.)

AuthDirFastGuarantee *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
Authoritative directories only. If non−zero, always vote the Fast flag for any relay advertising this amount of capacity or more. (Default: 100 KBytes)

AuthDirGuardBWGuarantee *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
Authoritative directories only. If non−zero, this advertised capacity or more is always sufficient to satisfy the bandwidth requirement for the Guard flag. (Default: 2 MBytes)

AuthDirHasIPv6Connectivity 0|1
Authoritative directories only. When set to 0, OR ports with an IPv6 address are not included in the authority's votes. When set to 1, IPv6 OR ports are tested for reachability like IPv4 OR ports. If the reachability test succeeds, the authority votes for the IPv6 ORPort, and votes Running for the relay. If the reachability test fails, the authority does not vote for the IPv6 ORPort, and does not vote Running (Default: 0)

The content of the consensus depends on the number of voting authorities that set AuthDirHasIPv6Connectivity:

If no authorities set AuthDirHasIPv6Connectivity 1, there will be no IPv6 ORPorts in the consensus.

If a minority of authorities set AuthDirHasIPv6Connectivity 1, unreachable IPv6 ORPorts will be removed from the consensus. But the majority of IPv4−only authorities will still vote the relay as Running. Reachable IPv6 ORPort lines will be included in the consensus

If a majority of voting authorities set AuthDirHasIPv6Connectivity 1, relays with unreachable IPv6 ORPorts will not be listed as Running. Reachable IPv6 ORPort lines will be included in the consensus (To ensure that any valid majority will vote relays with unreachable IPv6 ORPorts not Running, 75% of authorities must set AuthDirHasIPv6Connectivity 1.)

AuthDirInvalid *AddressPattern...*
Authoritative directories only. A set of address patterns for servers that will never be listed as "valid" in any network status document that this authority publishes.

AuthDirListBadExits 0|1
Authoritative directories only. If set to 1, this directory has some opinion about which nodes are unsuitable as exit nodes. (Do not set this to 1 unless you plan to list non−functioning exits as bad; otherwise, you are effectively voting in favor of every declared exit as an exit.)

AuthDirMaxServersPerAddr *NUM*
Authoritative directories only. The maximum number of servers that we will list as acceptable on a single IP address. Set this to "0" for "no limit". (Default: 2)

AuthDirPinKeys 0|1
Authoritative directories only. If non−zero, do not allow any relay to publish a descriptor if any other relay has reserved its <Ed25519,RSA> identity keypair. In all cases, Tor records every keypair it accepts in a journal if it is new, or if it differs from the most recently accepted pinning for one of the keys it contains. (Default: 1)

AuthDirReject *AddressPattern...*
Authoritative directories only. A set of address patterns for servers that will never be listed at all in any network status document that this authority publishes, or accepted as an OR address in any descriptor submitted for publication by this authority.

AuthDirBadExitCCs *CC,...*

AuthDirInvalidCCs *CC,...*

AuthDirRejectRequestsUnderLoad 0|1
If set, the directory authority will start rejecting directory requests from non relay connections by sending a 503 error code if it is under bandwidth pressure (reaching the configured limit if any). Relays will always tried to be answered even if this is on. (Default: 1)

AuthDirRejectCCs *CC,...*
Authoritative directories only. These options contain a comma−separated list of country codes such that any server in one of those country codes will be marked as a bad exit/invalid for use, or rejected entirely.

AuthDirSharedRandomness 0|1
Authoritative directories only. Switch for the shared random protocol. If zero, the authority won’t participate in the protocol. If non−zero (default), the flag "shared−rand−participate" is added to the authority vote indicating participation in the protocol. (Default: 1)

AuthDirTestEd25519LinkKeys 0|1
Authoritative directories only. If this option is set to 0, then we treat relays as "Running" if their RSA key is correct when we probe them, regardless of their Ed25519 key. We should only ever set this option to 0 if there is some major bug in Ed25519 link authentication that causes us to label all the relays as not Running. (Default: 1)

AuthDirTestReachability 0|1
Authoritative directories only. If set to 1, then we periodically check every relay we know about to see whether it is running. If set to 0, we vote Running for every relay, and don’t perform these tests. (Default: 1)

BridgePassword *Password*
If set, contains an HTTP authenticator that tells a bridge authority to serve all requested bridge information. Used by the (only partially implemented) "bridge community" design, where a community of bridge relay operators all use an alternate bridge directory authority, and their target user audience can periodically fetch the list of available community bridges to stay up−to−date. (Default: not set)

ConsensusParams *STRING*
STRING is a space−separated list of key=value pairs that Tor will include in the "params" line of its networkstatus vote. This directive can be specified multiple times so you don’t have to put it all on one line.

DirAllowPrivateAddresses 0|1
If set to 1, Tor will accept server descriptors with arbitrary "Address" elements. Otherwise, if the address is not an IP address or is a private IP address, it will reject the server descriptor. Additionally, Tor will allow exit policies for private networks to fulfill Exit flag requirements. (Default: 0)

GuardfractionFile *FILENAME*
V3 authoritative directories only. Configures the location of the guardfraction file which contains information about how long relays have been guards. (Default: unset)

MinMeasuredBWsForAuthToIgnoreAdvertised *N*
A total value, in abstract bandwidth units, describing how much measured total bandwidth an authority should have observed on the network before it will treat advertised bandwidths as wholly unreliable. (Default: 500)

MinUptimeHidServDirectoryV2 *N* **seconds|minutes|hours|days|weeks**
Minimum uptime of a relay to be accepted as a hidden service directory by directory authorities. (Default: 96 hours)

RecommendedClientVersions *STRING*
STRING is a comma−separated list of Tor versions currently believed to be safe for clients to use. This information is included in version 2 directories. If this is not set then the value of **RecommendedVersions** is used. When this is set then **VersioningAuthoritativeDirectory** should be set too.

RecommendedServerVersions *STRING*
STRING is a comma−separated list of Tor versions currently believed to be safe for servers to use. This information is included in version 2 directories. If this is not set then the value of **RecommendedVersions** is used. When this is set then **VersioningAuthoritativeDirectory** should be set too.

RecommendedVersions *STRING*
STRING is a comma−separated list of Tor versions currently believed to be safe. The list is included in each directory, and nodes which pull down the directory learn whether they need to upgrade. This option can appear multiple times: the values from multiple lines are spliced together. When this is set then **VersioningAuthoritativeDirectory** should be set too.

V3AuthDistDelay *N* **seconds|minutes|hours**
V3 authoritative directories only. Configures the server’s preferred delay between publishing its consensus and signature and assuming it has all the signatures from all the other authorities. Note that the actual time used is not the server’s preferred time, but the consensus of all preferences. (Default: 5 minutes)

V3AuthNIntervalsValid *NUM*
V3 authoritative directories only. Configures the number of VotingIntervals for which each consensus should be valid for. Choosing high numbers increases network partitioning risks; choosing low numbers increases directory traffic. Note that the actual number of intervals used is not the server’s preferred number, but the consensus of all preferences. Must be at least 2. (Default: 3)

V3AuthUseLegacyKey 0|1

If set, the directory authority will sign consensuses not only with its own signing key, but also with a "legacy" key and certificate with a different identity. This feature is used to migrate directory authority keys in the event of a compromise. (Default: 0)

V3AuthVoteDelay *N* **seconds**|**minutes**|**hours**
V3 authoritative directories only. Configures the server’s preferred delay between publishing its vote and assuming it has all the votes from all the other authorities. Note that the actual time used is not the server’s preferred time, but the consensus of all preferences. (Default: 5 minutes)

V3AuthVotingInterval *N* **minutes**|**hours**
V3 authoritative directories only. Configures the server’s preferred voting interval. Note that voting will *actually* happen at an interval chosen by consensus from all the authorities' preferred intervals. This time SHOULD divide evenly into a day. (Default: 1 hour)

V3BandwidthsFile *FILENAME*
V3 authoritative directories only. Configures the location of the bandwidth−authority generated file storing information on relays' measured bandwidth capacities. To avoid inconsistent reads, bandwidth data should be written to temporary file, then renamed to the configured filename. (Default: unset)

VersioningAuthoritativeDirectory 0|**1**
When this option is set to 1, Tor adds information on which versions of Tor are still believed safe for use to the published directory. Each version 1 authority is automatically a versioning authority; version 2 authorities provide this service optionally. See RecommendedVersions, RecommendedClientVersions, and RecommendedServerVersions.

HIDDEN SERVICE OPTIONS

The following options are used to configure a hidden service. Some options apply per service and some apply for the whole tor instance.

The next section describes the per service options that can only be set **after** the **HiddenServiceDir** directive

PER SERVICE OPTIONS:

HiddenServiceAllowUnknownPorts 0|**1**
If set to 1, then connections to unrecognized ports do not cause the current hidden service to close rendezvous circuits. (Setting this to 0 is not an authorization mechanism; it is instead meant to be a mild inconvenience to port−scanners.) (Default: 0)

HiddenServiceDir *DIRECTORY*
Store data files for a hidden service in DIRECTORY. Every hidden service must have a separate directory. You may use this option multiple times to specify multiple services. If DIRECTORY does not exist, Tor will create it. Please note that you cannot add new Onion Service to already running Tor instance if **Sandbox** is enabled. (Note: in current versions of Tor, if DIRECTORY is a relative path, it will be relative to the current working directory of Tor instance, not to its DataDirectory. Do not rely on this behavior; it is not guaranteed to remain the same in future versions.)

HiddenServiceDirGroupReadable 0|**1**
If this option is set to 1, allow the filesystem group to read the hidden service directory and hostname file. If the option is set to 0, only owner is able to read the hidden service directory. (Default: 0) Has no effect on Windows.

HiddenServiceEnableIntroDoSDefense 0|**1**
Enable DoS defense at the intropoint level. When this is enabled, the rate and burst parameter (see below) will be sent to the intro point which will then use them to apply rate limiting for introduction request to this service.

The introduction point honors the consensus parameters except if this is specifically set by the service operator using this option. The service never looks at the consensus parameters in order to enable or disable this defense. (Default: 0)

HiddenServiceEnableIntroDoSBurstPerSec *NUM*
The allowed client introduction burst per second at the introduction point. If this option is 0, it is considered infinite and thus if **HiddenServiceEnableIntroDoSDefense** is set, it then effectively disables the defenses. (Default: 200)

HiddenServiceEnableIntroDoSRatePerSec *NUM*
The allowed client introduction rate per second at the introduction point. If this option is 0, it is considered infinite and thus if **HiddenServiceEnableIntroDoSDefense** is set, it then effectively disables the defenses. (Default: 25)

HiddenServiceExportCircuitID *protocol*
The onion service will use the given protocol to expose the global circuit identifier of each inbound client circuit. The only protocol supported right now 'haproxy'. This option is only for v3 services. (Default: none)

The haproxy option works in the following way: when the feature is enabled, the Tor process will write a header line when a client is connecting to the onion service. The header will look like this:

"PROXY TCP6 fc00:dead:beef:4dad::ffff:ffff ::1 65535 42\r\n"

We encode the "global circuit identifier" as the last 32−bits of the first IPv6 address. All other values in the header can safely be ignored. You can compute the global circuit identifier using the following formula given the IPv6 address "fc00:dead:beef:4dad::AABB:CCDD":

global_circuit_id = (0xAA << 24) + (0xBB << 16) + (0xCC << 8) + 0xDD;

In the case above, where the last 32−bits are 0xffffffff, the global circuit identifier would be 4294967295. You can use this value together with Tor's control port to terminate particular circuits using their global circuit identifiers. For more information about this see control−spec.txt.

The HAProxy version 1 protocol is described in detail at https://www.haproxy.org/download/1.8/doc/proxy−protocol.txt

HiddenServiceOnionBalanceInstance 0|1
If set to 1, this onion service becomes an OnionBalance instance and will accept client connections destined to an OnionBalance frontend. In this case, Tor expects to find a file named "ob_config" inside the **HiddenServiceDir** directory with content:

MasterOnionAddress <frontend_onion_address>

where <frontend_onion_address> is the onion address of the OnionBalance frontend (e.g. wrxdvcaqpuzakbfww5sxs6r2uybczwijzfn2ezy2osaj7iox7kl7nhad.onion).

HiddenServiceMaxStreams *N*
The maximum number of simultaneous streams (connections) per rendezvous circuit. The maximum value allowed is 65535. (Setting this to 0 will allow an unlimited number of simultaneous streams.) (Default: 0)

HiddenServiceMaxStreamsCloseCircuit 0|1
If set to 1, then exceeding **HiddenServiceMaxStreams** will cause the offending rendezvous circuit to be torn down, as opposed to stream creation requests that exceed the limit being silently ignored.

(Default: 0)

HiddenServiceNumIntroductionPoints *NUM*
Number of introduction points the hidden service will have. You can't have more than 20. (Default: 3)

HiddenServicePort *VIRTPORT* [*TARGET*]
Configure a virtual port VIRTPORT for a hidden service. You may use this option multiple times; each time applies to the service using the most recent HiddenServiceDir. By default, this option maps the virtual port to the same port on 127.0.0.1 over TCP. You may override the target port, address, or both by specifying a target of addr, port, addr:port, or **unix:***path*. (You can specify an IPv6 target as [addr]:port. Unix paths may be quoted, and may use standard C escapes.) You may also have multiple lines with the same VIRTPORT: when a user connects to that VIRTPORT, one of the TARGETs from those lines will be chosen at random. Note that address−port pairs have to be comma−separated.

HiddenServiceVersion 3
A list of rendezvous service descriptor versions to publish for the hidden service. Currently, only version 3 is supported. (Default: 3)

PER INSTANCE OPTIONS:

HiddenServiceSingleHopMode 0|1
Experimental − Non Anonymous Hidden Services on a tor instance in HiddenServiceSingleHopMode make one−hop (direct) circuits between the onion service server, and the introduction and rendezvous points. (Onion service descriptors are still posted using 3−hop paths, to avoid onion service directories blocking the service.) This option makes every hidden service instance hosted by a tor instance a Single Onion Service. One−hop circuits make Single Onion servers easily locatable, but clients remain location−anonymous. However, the fact that a client is accessing a Single Onion rather than a Hidden Service may be statistically distinguishable.

WARNING: Once a hidden service directory has been used by a tor instance in HiddenServiceSingleHopMode, it can **NEVER** be used again for a hidden service. It is best practice to create a new hidden service directory, key, and address for each new Single Onion Service and Hidden Service. It is not possible to run Single Onion Services and Hidden Services from the same tor instance: they should be run on different servers with different IP addresses.

HiddenServiceSingleHopMode requires HiddenServiceNonAnonymousMode to be set to 1. Since a Single Onion service is non−anonymous, you can not configure a SOCKSPort on a tor instance that is running in **HiddenServiceSingleHopMode**. Can not be changed while tor is running. (Default: 0)

HiddenServiceNonAnonymousMode 0|1
Makes hidden services non−anonymous on this tor instance. Allows the non−anonymous HiddenServiceSingleHopMode. Enables direct connections in the server−side hidden service protocol. If you are using this option, you need to disable all client−side services on your Tor instance, including setting SOCKSPort to "0". Can not be changed while tor is running. (Default: 0)

PublishHidServDescriptors 0|1
If set to 0, Tor will run any hidden services you configure, but it won't advertise them to the rendezvous directory. This option is only useful if you're using a Tor controller that handles hidserv publishing for you. (Default: 1)

CLIENT AUTHORIZATION

Service side:

To configure client authorization on the service side, the "<HiddenServiceDir>/authorized_clients/" directory needs to exist. Each file in that directory should be suffixed with ".auth" (i.e. "alice.auth"; the file name is irrelevant) and its content format MUST be:

<auth−type>:<key−type>:<base32−encoded−public−key>

The supported <auth−type> are: "descriptor". The supported <key−type> are: "x25519". The <base32−encoded−public−key> is the base32 representation of the raw key bytes only (32 bytes for x25519).

Each file MUST contain one line only. Any malformed file will be ignored. Client authorization will only be enabled for the service if tor successfully loads at least one authorization file.

Note that once you've configured client authorization, anyone else with the address won't be able to access it from this point on. If no authorization is configured, the service will be accessible to anyone with the onion address.

Revoking a client can be done by removing their ".auth" file, however the revocation will be in effect only after the tor process gets restarted even if a SIGHUP takes place.

Client side:

To access a v3 onion service with client authorization as a client, make sure you have ClientOnionAuthDir set in your torrc. Then, in the <ClientOnionAuthDir> directory, create an .auth_private file for the onion service corresponding to this key (i.e. 'bob_onion.auth_private'). The contents of the <ClientOnionAuthDir>/<user>.auth_private file should look like:

<56−char−onion−addr−without−.onion−part>:descriptor:x25519:<x25519 private key in base32>

For more information, please see https://2019.www.torproject.org/docs/tor−onion−service.html.en#ClientAuthorization .

TESTING NETWORK OPTIONS

The following options are used for running a testing Tor network.

TestingTorNetwork 0|1

If set to 1, Tor adjusts default values of the configuration options below, so that it is easier to set up a testing Tor network. May only be set if non−default set of DirAuthorities is set. Cannot be unset while Tor is running. (Default: 0)

DirAllowPrivateAddresses 1
EnforceDistinctSubnets 0
AuthDirMaxServersPerAddr 0
ClientBootstrapConsensusAuthorityDownloadInitialDelay 0
ClientBootstrapConsensusFallbackDownloadInitialDelay 0
ClientBootstrapConsensusAuthorityOnlyDownloadInitialDelay 0
ClientDNSRejectInternalAddresses 0
ClientRejectInternalAddresses 0
CountPrivateBandwidth 1
ExitPolicyRejectPrivate 0
ExtendAllowPrivateAddresses 1
V3AuthVotingInterval 5 minutes
V3AuthVoteDelay 20 seconds
V3AuthDistDelay 20 seconds
TestingV3AuthInitialVotingInterval 150 seconds
TestingV3AuthInitialVoteDelay 20 seconds

TestingV3AuthInitialDistDelay 20 seconds
TestingAuthDirTimeToLearnReachability 0 minutes
MinUptimeHidServDirectoryV2 0 minutes
TestingServerDownloadInitialDelay 0
TestingClientDownloadInitialDelay 0
TestingServerConsensusDownloadInitialDelay 0
TestingClientConsensusDownloadInitialDelay 0
TestingBridgeDownloadInitialDelay 10
TestingBridgeBootstrapDownloadInitialDelay 0
TestingClientMaxIntervalWithoutRequest 5 seconds
TestingDirConnectionMaxStall 30 seconds
TestingEnableConnBwEvent 1
TestingEnableCellStatsEvent 1
RendPostPeriod 2 minutes

TestingAuthDirTimeToLearnReachability *N* **seconds**|**minutes**|**hours**
After starting as an authority, do not make claims about whether routers are Running until this much time has passed. Changing this requires that **TestingTorNetwork** is set. (Default: 30 minutes)

TestingAuthKeyLifetime *N* **seconds**|**minutes**|**hours**|**days**|**weeks**|**months**
Overrides the default lifetime for a signing Ed25519 TLS Link authentication key. (Default: 2 days)

TestingAuthKeySlop *N* **seconds**|**minutes**|**hours**

TestingBridgeBootstrapDownloadInitialDelay *N*
Initial delay in seconds for when clients should download each bridge descriptor when they have just started, or when they can not contact any of their bridges. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingBridgeDownloadInitialDelay *N*
Initial delay in seconds for when clients should download each bridge descriptor when they know that one or more of their configured bridges are running. Changing this requires that **TestingTorNetwork** is set. (Default: 10800)

TestingClientConsensusDownloadInitialDelay *N*
Initial delay in seconds for when clients should download consensuses. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingClientDownloadInitialDelay *N*
Initial delay in seconds for when clients should download things in general. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingClientMaxIntervalWithoutRequest *N* **seconds**|**minutes**
When directory clients have only a few descriptors to request, they batch them until they have more, or until this amount of time has passed. Changing this requires that **TestingTorNetwork** is set. (Default: 10 minutes)

TestingDirAuthVoteExit *node*,*node*,...
A list of identity fingerprints, country codes, and address patterns of nodes to vote Exit for regardless of their uptime, bandwidth, or exit policy. See ExcludeNodes for more information on how to specify nodes.

In order for this option to have any effect, **TestingTorNetwork** has to be set. See ExcludeNodes for more information on how to specify nodes.

TestingDirAuthVoteExitIsStrict 0|**1**
If True (1), a node will never receive the Exit flag unless it is specified in the **TestingDirAuthVoteExit** list, regardless of its uptime, bandwidth, or exit policy.

In order for this option to have any effect, **TestingTorNetwork** has to be set.

TestingDirAuthVoteGuard *node,node,...*
A list of identity fingerprints and country codes and address patterns of nodes to vote Guard for regardless of their uptime and bandwidth. See ExcludeNodes for more information on how to specify nodes.

In order for this option to have any effect, **TestingTorNetwork** has to be set.

TestingDirAuthVoteGuardIsStrict 0|1
If True (1), a node will never receive the Guard flag unless it is specified in the **TestingDirAuthVoteGuard** list, regardless of its uptime and bandwidth.

In order for this option to have any effect, **TestingTorNetwork** has to be set.

TestingDirAuthVoteHSDir *node,node,...*
A list of identity fingerprints and country codes and address patterns of nodes to vote HSDir for regardless of their uptime and DirPort. See ExcludeNodes for more information on how to specify nodes.

In order for this option to have any effect, **TestingTorNetwork** must be set.

TestingDirAuthVoteHSDirIsStrict 0|1
If True (1), a node will never receive the HSDir flag unless it is specified in the **TestingDirAuthVoteHSDir** list, regardless of its uptime and DirPort.

In order for this option to have any effect, **TestingTorNetwork** has to be set.

TestingDirConnectionMaxStall *N* **seconds|minutes**
Let a directory connection stall this long before expiring it. Changing this requires that **TestingTorNetwork** is set. (Default: 5 minutes)

TestingEnableCellStatsEvent 0|1
If this option is set, then Tor controllers may register for CELL_STATS events. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingEnableConnBwEvent 0|1
If this option is set, then Tor controllers may register for CONN_BW events. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingLinkCertLifetime *N* **seconds|minutes|hours|days|weeks|months**
Overrides the default lifetime for the certificates used to authenticate our X509 link cert with our ed25519 signing key. (Default: 2 days)

TestingLinkKeySlop *N* **seconds|minutes|hours**

TestingMinExitFlagThreshold *N* **KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
Sets a lower−bound for assigning an exit flag when running as an authority on a testing network. Overrides the usual default lower bound of 4 KBytes. (Default: 0)

TestingMinFastFlagThreshold *N* **bytes|KBytes|MBytes|GBytes|TBytes|KBits|MBits|GBits|TBits**
Minimum value for the Fast flag. Overrides the ordinary minimum taken from the consensus when TestingTorNetwork is set. (Default: 0.)

TestingMinTimeToReportBandwidth *N* **seconds|minutes|hours**
Do not report our measurements for our maximum observed bandwidth for any time period that has lasted for less than this amount of time. Values over 1 day have no effect. (Default: 1 day)

TestingServerConsensusDownloadInitialDelay *N*
Initial delay in seconds for when servers should download consensuses. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingServerDownloadInitialDelay *N*
Initial delay in seconds for when servers should download things in general. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

TestingSigningKeySlop *N* **seconds**|**minutes**|**hours**
How early before the official expiration of a an Ed25519 signing key do we replace it and issue a new key? (Default: 3 hours for link and auth; 1 day for signing.)

TestingV3AuthInitialDistDelay *N* **seconds**|**minutes**|**hours**
Like V3AuthDistDelay, but for initial voting interval before the first consensus has been created. Changing this requires that **TestingTorNetwork** is set. (Default: 5 minutes)

TestingV3AuthInitialVoteDelay *N* **seconds**|**minutes**|**hours**
Like V3AuthVoteDelay, but for initial voting interval before the first consensus has been created. Changing this requires that **TestingTorNetwork** is set. (Default: 5 minutes)

TestingV3AuthInitialVotingInterval *N* **seconds**|**minutes**|**hours**
Like V3AuthVotingInterval, but for initial voting interval before the first consensus has been created. Changing this requires that **TestingTorNetwork** is set. (Default: 30 minutes)

TestingV3AuthVotingStartOffset *N* **seconds**|**minutes**|**hours**
Directory authorities offset voting start time by this much. Changing this requires that **TestingTorNetwork** is set. (Default: 0)

NON−PERSISTENT OPTIONS

These options are not saved to the torrc file by the "SAVECONF" controller command. Other options of this type are documented in control−spec.txt, section 5.4. End−users should mostly ignore them.

__ControlPort, **__DirPort**, **__DNSPort**, **__ExtORPort**, **__NATDPort**, **__ORPort**, **__SocksPort**, **__TransPort**
These underscore−prefixed options are variants of the regular Port options. They behave the same, except they are not saved to the torrc file by the controller's SAVECONF command.

SIGNALS

Tor catches the following signals:

SIGTERM
Tor will catch this, clean up and sync to disk if necessary, and exit.

SIGINT
Tor clients behave as with SIGTERM; but Tor servers will do a controlled slow shutdown, closing listeners and waiting 30 seconds before exiting. (The delay can be configured with the ShutdownWaitLength config option.)

SIGHUP
The signal instructs Tor to reload its configuration (including closing and reopening logs), and kill and restart its helper processes if applicable.

SIGUSR1
Log statistics about current connections, past connections, and throughput.

SIGUSR2
Switch all logs to loglevel debug. You can go back to the old loglevels by sending a SIGHUP.

SIGCHLD
Tor receives this signal when one of its helper processes has exited, so it can clean up.

SIGPIPE
Tor catches this signal and ignores it.

SIGXFSZ

If this signal exists on your platform, Tor catches and ignores it.

FILES

/etc/tor/torrc
Default location of the configuration file.

$HOME/.torrc
Fallback location for torrc, if /etc/tor/torrc is not found.

/var/lib/tor/
The tor process stores keys and other data here.

CacheDirectory/**cached−certs**
Contains downloaded directory key certificates that are used to verify authenticity of documents generated by the Tor directory authorities.

CacheDirectory/**cached−consensus** and/or **cached−microdesc−consensus**
The most recent consensus network status document we've downloaded.

CacheDirectory/**cached−descriptors** and **cached−descriptors.new**
These files contain the downloaded router statuses. Some routers may appear more than once; if so, the most recently published descriptor is used. Lines beginning with **@**−signs are annotations that contain more information about a given router. The **.new** file is an append−only journal; when it gets too large, all entries are merged into a new cached−descriptors file.

CacheDirectory/**cached−extrainfo** and **cached−extrainfo.new**
Similar to **cached−descriptors**, but holds optionally−downloaded "extra−info" documents. Relays use these documents to send inessential information about statistics, bandwidth history, and network health to the authorities. They aren't fetched by default. See DownloadExtraInfo for more information.

CacheDirectory/**cached−microdescs** and **cached−microdescs.new**
These files hold downloaded microdescriptors. Lines beginning with **@**−signs are annotations that contain more information about a given router. The **.new** file is an append−only journal; when it gets too large, all entries are merged into a new cached−microdescs file.

DataDirectory/**state**
Contains a set of persistent key−value mappings. These include:

- the current entry guards and their status.
- the current bandwidth accounting values.
- when the file was last written
- what version of Tor generated the state file
- a short history of bandwidth usage, as produced in the server descriptors.

DataDirectory/**sr−state**
Authority only. This file is used to record information about the current status of the shared−random−value voting state.

CacheDirectory/**diff−cache**
Directory cache only. Holds older consensuses and diffs from oldest to the most recent consensus of each type compressed in various ways. Each file contains a set of key−value arguments describing its contents, followed by a single NUL byte, followed by the main file contents.

DataDirectory/**bw_accounting**
This file is obsolete and the data is now stored in the **state** file instead. Used to track bandwidth accounting values (when the current period starts and ends; how much has been read and written so far this period).

DataDirectory/**control_auth_cookie**
This file can be used only when cookie authentication is enabled. Used for cookie authentication with the controller. Location can be overridden by the CookieAuthFile configuration option. Regenerated

on startup. See control−spec.txt in torspec for details.

DataDirectory/**lock**
This file is used to prevent two Tor instances from using the same data directory. If access to this file is locked, data directory is already in use by Tor.

DataDirectory/**key−pinning−journal**
Used by authorities. A line−based file that records mappings between RSA1024 and Ed25519 identity keys. Authorities enforce these mappings, so that once a relay has picked an Ed25519 key, stealing or factoring the RSA1024 key will no longer let an attacker impersonate the relay.

KeyDirectory/**authority_identity_key**
A v3 directory authority's master identity key, used to authenticate its signing key. Tor doesn't use this while it's running. The tor−gencert program uses this. If you're running an authority, you should keep this key offline, and not put it in this file.

KeyDirectory/**authority_certificate**
Only directory authorities use this file. A v3 directory authority's certificate which authenticates the authority's current vote− and consensus−signing key using its master identity key.

KeyDirectory/**authority_signing_key**
Only directory authorities use this file. A v3 directory authority's signing key that is used to sign votes and consensuses. Corresponds to the **authority_certificate** cert.

KeyDirectory/**legacy_certificate**
As authority_certificate; used only when V3AuthUseLegacyKey is set. See documentation for V3AuthUseLegacyKey.

KeyDirectory/**legacy_signing_key**
As authority_signing_key: used only when V3AuthUseLegacyKey is set. See documentation for V3AuthUseLegacyKey.

KeyDirectory/**secret_id_key**
A relay's RSA1024 permanent identity key, including private and public components. Used to sign router descriptors, and to sign other keys.

KeyDirectory/**ed25519_master_id_public_key**
The public part of a relay's Ed25519 permanent identity key.

KeyDirectory/**ed25519_master_id_secret_key**
The private part of a relay's Ed25519 permanent identity key. This key is used to sign the medium−term ed25519 signing key. This file can be kept offline or encrypted. If so, Tor will not be able to generate new signing keys automatically; you'll need to use tor −−keygen to do so.

KeyDirectory/**ed25519_signing_secret_key**
The private and public components of a relay's medium−term Ed25519 signing key. This key is authenticated by the Ed25519 master key, which in turn authenticates other keys (and router descriptors).

KeyDirectory/**ed25519_signing_cert**
The certificate which authenticates "ed25519_signing_secret_key" as having been signed by the Ed25519 master key.

KeyDirectory/**secret_onion_key** and **secret_onion_key.old**
A relay's RSA1024 short−term onion key. Used to decrypt old−style ("TAP") circuit extension requests. The **.old** file holds the previously generated key, which the relay uses to handle any requests that were made by clients that didn't have the new one.

KeyDirectory/**secret_onion_key_ntor** and **secret_onion_key_ntor.old**
A relay's Curve25519 short−term onion key. Used to handle modern ("ntor") circuit extension requests. The **.old** file holds the previously generated key, which the relay uses to handle any requests that were made by clients that didn't have the new one.

DataDirectory/**fingerprint**
Only used by servers. Contains the fingerprint of the server's identity key.

DataDirectory/**hashed−fingerprint**
Only used by bridges. Contains the hashed fingerprint of the bridge's identity key. (That is, the hash of the hash of the identity key.)

DataDirectory/**approved−routers**
Only used by authoritative directory servers. Each line lists a status and an identity, separated by whitespace. Identities can be hex−encoded RSA fingerprints, or base−64 encoded ed25519 public keys. See the **fingerprint** file in a tor relay's *DataDirectory* for an example fingerprint line. If the status is **!reject**, then descriptors from the given identity are rejected by this server. If it is **!invalid** then descriptors are accepted, but marked in the vote as not valid. If it is **!badexit**, then the authority will vote for it to receive a BadExit flag, indicating that it shouldn't be used for traffic leaving the Tor network. (Neither rejected nor invalid relays are included in the consensus.)

DataDirectory/**v3−status−votes**
Only for v3 authoritative directory servers. This file contains status votes from all the authoritative directory servers.

CacheDirectory/**unverified−consensus**
Contains a network consensus document that has been downloaded, but which we didn't have the right certificates to check yet.

CacheDirectory/**unverified−microdesc−consensus**
Contains a microdescriptor−flavored network consensus document that has been downloaded, but which we didn't have the right certificates to check yet.

DataDirectory/**unparseable−desc**
Onion server descriptors that Tor was unable to parse are dumped to this file. Only used for debugging.

DataDirectory/**router−stability**
Only used by authoritative directory servers. Tracks measurements for router mean−time−between−failures so that authorities have a fair idea of how to set their Stable flags.

DataDirectory/**stats/dirreq−stats**
Only used by directory caches and authorities. This file is used to collect directory request statistics.

DataDirectory/**stats/entry−stats**
Only used by servers. This file is used to collect incoming connection statistics by Tor entry nodes.

DataDirectory/**stats/bridge−stats**
Only used by servers. This file is used to collect incoming connection statistics by Tor bridges.

DataDirectory/**stats/exit−stats**
Only used by servers. This file is used to collect outgoing connection statistics by Tor exit routers.

DataDirectory/**stats/buffer−stats**
Only used by servers. This file is used to collect buffer usage history.

DataDirectory/**stats/conn−stats**
Only used by servers. This file is used to collect approximate connection history (number of active connections over time).

DataDirectory/**stats/hidserv−stats**
Only used by servers. This file is used to collect approximate counts of what fraction of the traffic is hidden service rendezvous traffic, and approximately how many hidden services the relay has seen.

DataDirectory/**networkstatus−bridges'**
Only used by authoritative bridge directories. Contains information about bridges that have self−reported themselves to the bridge authority.

HiddenServiceDirectory/**hostname**
The <base32−encoded−fingerprint>.onion domain name for this hidden service. If the hidden service

is restricted to authorized clients only, this file also contains authorization data for all clients.

Note

The clients will ignore any extra subdomains prepended to a hidden service hostname. Supposing you have "xyz.onion" as your hostname, you can ask your clients to connect to "www.xyz.onion" or "irc.xyz.onion" for virtual−hosting purposes.

HiddenServiceDirectory/**private_key**
Contains the private key for this hidden service.

HiddenServiceDirectory/**client_keys**
Contains authorization data for a hidden service that is only accessible by authorized clients.

HiddenServiceDirectory/**onion_service_non_anonymous**
This file is present if a hidden service key was created in **HiddenServiceNonAnonymousMode**.

SEE ALSO

For more information, refer to the Tor Project website at https://www.torproject.org/ and the Tor specifications at https://spec.torproject.org. See also **torsocks**(1) and **torify**(1).

BUGS

Because Tor is still under development, there may be plenty of bugs. Please report them at https://bugs.torproject.org/.