Generated by sbws version 1.0.3

```
1523911758
version=1.2.0
latest_bandwidth=2018-04-16T20:49:18
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=6000
percent_eligible_relays=93
software=sbws
software_version=1.0.3
=====

bw=38000 bw_mean=1127824 bw_median=1180062 desc_bw_avg=1073741824
desc_bw_obs_last=17230879 desc_bw_obs_mean=14732306 error_circ=0
error_misc=0 error_stream=1
master_key_ed25519=YaqV4vbvPYKucElk297eVdNArDz9HtIwUoIeo0+cVIpQ
nick=Test node_id=$68A483E05A2ABDCA6DA5A3EF8DB5177638A27F80 rtt=380
success=1 time=2018-05-08T16:13:26
bw=1 bw_mean=199162 bw_median=185675 desc_bw_avg=409600
desc_bw_obs_last=836165 desc_bw_obs_mean=858030 error_circ=0
error_misc=0 error_stream=0
master_key_ed25519=a6a+dZadrQBtfSbmQkP7j2ardCmLnm5NJ4ZzkvDxbo0I
nick=Test2 node_id=$96C15995F30895689291F455587BD94CA427B6FC rtt=378
success=1 time=2018-05-08T16:13:36
``
```


When there are not enough eligible measured relays { #sbws-103-not-enough-measured }

```text
1540496079
version=1.2.0
earliest_bandwidth=2018-10-20T19:35:52
file_created=2018-10-25T19:35:03
generator_started=2018-10-25T11:42:56
latest_bandwidth=2018-10-25T19:34:39
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=2960
percent_eligible_relays=46
software=sbws
software_version=1.0.3
=====
```

Headers generated by sbws version 1.0.4

```
1523911758
version=1.2.0
latest_bandwidth=2018-04-16T20:49:18
destinations_countries=TH,ZZ
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=6000
percent_eligible_relays=93
scanner_country=SN
software=sbws
software_version=1.0.4
=====
```

https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/attic/rend-spec-v2.txt?ref_type=heads (Tor Onion Service Rendezvous Specification, Version 2 (Obsolete))

/rend-spec-v3

https://spec.torproject.org/rend-spec (Tor Onion Service Rendezvous Specification, Version 3 (Latest))

/socks-extensions

https://spec.torproject.org/socks-extensions (Tor's extensions to the SOCKS protocol)

/srv-spec

https://spec.torproject.org/srv-spec (Tor Shared Random Subsystem Specification)

/tor-fw-helper-spec

https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/attic/tor-fw-helper-spec.txt?ref_type=heads (Tor's (little) Firewall Helper specification)

/tor-spec

https://spec.torproject.org/tor-spec (Tor Protocol Specification)

/torbrowser-design

https://2019.www.torproject.org/projects/torbrowser/design/ (The Design and Implementation of the Tor Browser)

/version-spec

https://spec.torproject.org/version-spec (How Tor Version Numbers Work)

/tor-design

https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf (Tor: The Second-Generation Onion Router)

/walking-onions

https://spec.torproject.org/proposals/323-walking-onions-full.html (Walking Onions specifications)

Permalinks

These URLs at `spec.toprorject.org` are intended to be long-term permalinks.

/address-spec
https://spec.torproject.org/address-spec (Special Hostnames in Tor)
/bandwidth-file-spec
https://spec.torproject.org/bandwidth-file-spec (Directory Authority Bandwidth File spec)
/bridgedb-spec
https://spec.torproject.org/bridgedb-spec (BridgeDB specification)
/cert-spec
https://spec.torproject.org/cert-spec (Ed25519 certificates in Tor)
/collector-protocol
https://gitlab.torproject.org/tpo/network-health/metrics/collector/-/blob/master/src/main/resources/docs/PROTOCOL?ref_type=heads (Protocol of CollecTor's File Structure)
/control-spec
https://spec.torproject.org/control-spec (Tor control protocol, version 1)
/dir-spec
https://spec.torproject.org/dir-spec (Tor directory protocol, version 3)
/dir-list-spec
https://spec.torproject.org/dir-list-spec (Tor Directory List file format)
/ext-orport-spec
https://spec.torproject.org/ext-orport-spec (Extended ORPort for pluggable transports)
/gettor-spec
https://gitlab.torproject.org/tpo/core/torspec/-/raw/main/attic/text_formats/gettor-spec.txt?ref_type=heads (GetTor specification)
/padding-spec
https://spec.torproject.org/padding-spec (Tor Padding Specification)
/path-spec
https://spec.torproject.org/path-spec (Tor Path Specification)
/pt-spec
https://spec.torproject.org/pt-spec (Tor Pluggable Transport Specification, version 1)
/rend-spec
https://spec.torproject.org/rend-spec (Tor Onion Service Rendezvous Specification, latest version)
/rend-spec-v2

Generated by sbws version 1.1.0

```
1523911758
version=1.4.0
latest_bandwidth=2018-04-16T20:49:18
destinations_countries=TH,ZZ
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=6000
percent_eligible_relays=93
recent_measurement_attempt_count=6243
recent_measurement_failure_count=732
recent_measurements_excluded_error_count=969
recent_measurements_excluded_few_count=3946
recent_measurements_excluded_near_count=90
recent_measurements_excluded_old_count=0
recent_priority_list_count=20
recent_priority_relay_count=6243
scanner_country=SN
software=sbws
software_version=1.1.0
time_to_report_half_network=57273
=====

bw=1 error_circ=1 error_destination=0 error_misc=0
error_second_relay=0 error_stream=0
master_key_ed25519=J3HQ24kOQWac3L1xlFLp7gY91qkb5NuKxjj1BhDi+m8
nick=snap269 node_id=$DC4D609F95A52614D1E69C752168AF1FCAE0B05F
relay_recent_measurement_attempt_count=3
relay_recent_measurements_excluded_error_count=1
relay_recent_measurements_excluded_near_count=3
relay_recent_consensus_count=3 relay_recent_priority_list_count=3
success=3 time=2019-03-16T18:20:57 unmeasured=1 vote=0
bw=1 error_circ=0 error_destination=0 error_misc=0
error_second_relay=0 error_stream=2
master_key_ed25519=h6ZB1E1yBFWIMloUm9IWwjgaPXEpL5cUbuoQDgdSDKg
nick=relay node_id=$C4544F9E209A9A9B99591D548B3E2822236C0503
relay_recent_measurement_attempt_count=3
relay_recent_measurements_excluded_error_count=2
relay_recent_measurements_excluded_few_count=1
relay_recent_consensus_count=3 relay_recent_priority_list_count=3
success=1 time=2019-03-17T06:50:58 unmeasured=1 vote=0
```

Scaling bandwidths

Scaling requirements

```
Tor accepts zero bandwidths, but they trigger bugs in older Tor
implementations. Therefore, scaling methods SHOULD perform the
following checks:
 * If the total bandwidth is zero, all relays should be given equal
 bandwidths.
 * If the scaled bandwidth is zero, it should be rounded up to one.
```

Initial experiments indicate that scaling may not be needed for torflow and sbws, because their measured bandwidths are similar enough already.

A linear scaling method

If scaling is required, here is a simple linear bandwidth scaling method, which ensures that all bandwidth votes contain approximately the same total bandwidth:

```
  1. Calculate the relay quota by dividing the total measured
bandwidth
     in all votes, by the number of relays with measured bandwidth
     votes. In the public tor network, this is approximately 7500 as
of
     April 2018. The quota should be a consensus parameter, so it can
be
     adjusted for all generators on the network.

  2. Calculate a vote quota by multiplying the relay quota by the
number
     of relays this bandwidth authority has measured
     bandwidths for.

  3. Calculate a scaling factor by dividing the vote quota by the
     total unscaled measured bandwidth in this bandwidth
     authority's upcoming vote.

  4. Multiply each unscaled measured bandwidth by the scaling
     factor.
```

Now, the total scaled bandwidth in the upcoming vote is approximately equal to the quota.

If you need to change a heading, make sure that you keep its `id` the same as it was before, so that links will still work.

Finally, when you're looking for specific sections (e.g., to fix references that say "See section 5.2.3") you can look for the HTML anchors that our conversion process added. For example, if you want to find `dir-spec.txt` section 2.1.3, look for the anchor that says `<a id="dir-spec.txt-2.1.3"></a>`.

Specific Markdown syntax

Define link fragment targets with `<span id="...">` (usually)

To manually make an target for a `#`-prefixed link fragment, prefer `<span id="fragment">Text</span>`, to `<a id=...>Text</a>`. This works around mdbook mistakenly styling `<a>` without `href` as if it were a clickable link.

(Of course it is often better to make the referenced text a section and use the mdbook explicit anchor syntax.)

You may need to make sure you have some other text on the same line as the `<span>` to avoid mdbook thinking you were writing a whole paragraph of raw HTML.

Sometimes you may wish to use `<div id="...">` and `</div>` (which must usually be placed as paragraphs of their own).

Example:

```
<span id="lemons-lemonade">LEMONS
are a sour fruit, sometimes used for making
lemonade</span>

(later)

Various fruit may be found, including [lemons](#lemons-lemonade).
```

It is OK to use an *empty* `a` element: `<a id=....></a>`. Many existing section anchors are done this way.

Documenting keys

TODO: Explain our new key documentation conventions, as used here.

Documentating data encodings

We have two competing legacy schemes for documenting data encodings. One of them is an ad-hoc format the looks like this:

```
    * FIELD_1     [4 bytes]
    * FIELD_2     [1 byte]
```

The other is a somewhat C-like format based on the `trunnel` format. It looks like this:

```
 struct message {
    u32 field_1;
    u8 field_2;
 }
```

Neither of these is really great. We should find something better.

Writing explanations

> When you are writing an explanation in the middle of a bunch of normative text, it is a good idea to put it in quoted text, like this.

Managing links

We're in the early stages of this spec organization, but we should still be thinking about long term maintainability.

Please think about how to keep links working in the long term. If you are going to add a link to a file, make sure that the file's name is reasonable. Before you rename a file, consider adding a redirect from the file's old name. (See the mdbook documentation for more information about how.)

If you want to link to a specific section within a file, make sure that the section has a defined anchor that makes sense. The syntax to define heading ids in mdbook looks like this:

```
## Heading with a long title that you want shorter name for
{ #shortname }
```

Quota changes

If all generators are using scaling, the quota can be gradually reduced or increased as needed. Smaller quotas decrease the size of uncompressed consensuses, and may decrease the size of consensus diffs and compressed consensuses. But if the relay quota is too small, some relays may be over- or under-weighted.

Torflow aggregation

Torflow implements two methods to compute the bandwidth values from the (stream) bandwidth measurements: with and without PID control feedback. The method described here is without PID control (see Torflow specification, section 2.2).

In the following sections, the relays' measured bandwidth refer to the ones that this bandwidth authority has measured for the relays that would be included in the next bandwidth authority's upcoming vote.

```
  1. Calculate the filtered bandwidth for each relay:
    - choose the relay's measurements (`bw_j`) that are equal or
greater
      than the mean of the measurements for this relay
    - calculate the mean of those measurements

    In pseudocode:

      bw_filt_i = mean(max(mean(bw_j), bw_j))

  2. Calculate network averages:
    - calculate the filtered average by dividing the sum of all the
      relays' filtered bandwidth by the number of relays that have
been
      measured (`n`), ie, calculate the mean average of the relays'
      filtered bandwidth.
    - calculate the stream average by dividing the sum of all the
      relays' measured bandwidth by the number of relays that have
been
      measured (`n`), ie, calculate the mean average or the relays'
      measured bandwidth.

     In pseudocode:

       bw_avg_filt_ = bw_filt_i / n
       bw_avg_strm = bw_i / n

  3. Calculate ratios for each relay:
    - calculate the filtered ratio by dividing each relay filtered
      bandwidth by the filtered average
    - calculate the stream ratio by dividing each relay measured
      bandwidth by the stream average

    In pseudocode:
```

r_filt_i = bw_filt_i / bw_avg_filt r_strm_i = bw_i / bw_avg_strm

- Tor project glossary
- Specifications glossary

(Please add more if you know about them!)

As we refine the guidelines in this file, we should attempt to get them upstreamed to more project-wide guides, if they are suitable.

Line breaks

Begin each new thought, sentence, or subclause on its own line. Keep lines short enough to be readable on a terminal (~80 columns), but don't reflow text unnecessarily. This makes diffs easier to review, while still keeping the text fairly readable in its raw form. See semantic linefeeds for history and a more detailed explanation of why this is useful.

Vocabulary

We use these terms freely:

- Channel
- Circuit
- Stream

We try not to say "connection" without qualification: There are too many things in Tor that can be called a "connection".

Similarly, don't say "session" without qualification except when it is clear from context what we mean.

Prefer "relay" to "node" or "server".

Prefer "service" and "client" when talking about onion services and their users.

Refer to arti as arti and the C tor implementation as "the C tor implementation" on first mention. Subsequently you can call it `tor` or "C tor".

Avoid "AP" and "OP" and "OR"; those are not in current usage.

The functions SHA-1, SHA-256, and SHA3-256 are called "hash functions". A "digest" is the output of a hash function.

Tor specifications: style and usage notes

Audience

The primary intended audiences are:

- Programmers: implementors of the Tor Protocols. This includes both maintainers of existing implementations, and people writing new implementations.
- Researchers: people analysing the security of the Tor Protocols, including both academic research and practical security investigation.
- Expert users who wish to fully understand the operation of their Tor software. This includes users of clients and relays.
- Directory authority operators, and others with a critical technical role in the integrity of the Tor network.

Scope and intentions

These notes apply to our specifications. When possible, they should also apply to proposals, to make proposals easier to merge into our specifications when they are done.

As of 2023, our existing specifications have been written without any real style guidelines, so you should not expect to find these guidelines to apply well to all of the documents as you read them. Instead, these guidelines are for documentation going forward.

These notes are not terribly well organized. We should improve them over time. They are meant to be a living document.

Other sources

There are a number of other style guides used in Tor. We should follow these as well. If they do not suit our needs, we should try to get them changed.

- Community team guidelines

```
  4. Calculate the final ratio for each relay:
    The final ratio is the larger between the filtered bandwidth's and
the
    stream bandwidth's ratio.

    In pseudocode:

      r_i = max(r_filt_i, r_strm_i)

  5. Calculate the scaled bandwidth for each relay:
    The most recent descriptor observed bandwidth (`bw_obs_i`) is
    multiplied by the ratio

    In pseudocode:

      bw_new_i = r_i * bw_obs_i
```

<<In this way, the resulting network status consensus bandwidth values are effectively re-weighted proportional to how much faster the node was as compared to the rest of the network.>>

Tor Directory List Format

Tim Wilson-Brown (teor)

Scope and Preliminaries

This document describes the format of Tor's directory lists, which are compiled and hard-coded into the tor binary. There is currently one list: the fallback directory mirrors. This list is also parsed by other libraries, like stem and metrics-lib. Alternate Tor implementations can use this list to bootstrap from the latest public Tor directory information.

The FallbackDir feature was introduced by proposal 210, and was first supported by Tor in Tor version 0.2.4.7-alpha. The first hard-coded list was shipped in 0.2.8.1-alpha.

The hard-coded fallback directory list is located in the tor source repository at:

`src/app/config/fallback_dirs.inc`

In Tor 0.3.4 and earlier, the list is located at:

`src/or/fallback_dirs.inc`

This document describes version 2.0.0 and later of the directory list format.

Legacy, semi-structured versions of the fallback list were released with Tor 0.2.8.1-alpha through Tor 0.3.1.9. We call this format version 1. Stem and Relay Search have parsers for this legacy format.

Format Overview

A directory list is a C code fragment containing an array of C string constants. Each double-quoted C string constant is a valid torrc FallbackDir entry. Each entry contains various data fields.

Directory lists do not include the C array's declaration, or the array's terminating NULL. Entries in directory lists do not include the FallbackDir torrc option. These are handled by the including C code.

Each book's source files are listed, and the chapter defined, in its `SUMMARY.md`. The format is pretty restrictive; see the mdbook documentation.

Editing advice

To edit these specs, clone the git repository and edit the appropriate file in the `spec` directory. These files will match the URLs of their corresponding pages, so if you want to edit `tor-spec/flow-control.html`, you'll be looking for a file called `spec/tor-spec/flow-control.md`.

We have started a style guide for writing new parts of this spec; as of 2023 it is quite preliminary. You should feel free to edit it!

About the Tor Specifications documents

The canonical, official, versions of these documents are on the Tor Specifications website maintained by the Tor Project.

Only the Tor Specifications themselves are approved. The Proposals are, by their nature, drafts.

When linking to the Specifications, consider using one of the links advertised in the Table of Permalinks.

Source code

The Specifications and Proposals are maintained by the Tor Project in a gitlab repository.

Corrections and clarifications are welcome. To propose a change to the Tor protocol, use the Proposals process

Building

The documents are in Markdown and formatted with mdbook. To build the formatted HTML:

```
cargo install mdbook
git clone https://gitlab.torproject.org/tpo/core/torspec/
cd torspec
bin/build_html
```

The output is then in `html/`.

Source code structure, and output webtree

There are two mdbook books here:

- **The Tor Specifications**: source code in `specs/`, formatted output in `html/`.
- **Proposals**: source code in `proposals/`, formatted output in `html/proposals/`.

Directory lists also include C-style comments and whitespace. The presence of whitespace may be significant, but the amount of whitespace is never significant. The type of whitespace is not significant to the C compiler or Tor C string parser. However, other parsers MAY rely on the distinction between newlines and spaces. (And that the only whitespace characters in the list are newlines and spaces.)

The directory entry C string constants are split over multiple lines for readability. Structured C-style comments are used to provide additional data fields. This information is not used by Tor, but may be of interest to other libraries.

The order of directory entries and data fields is not significant, except where noted below.

Acknowledgements

The original fallback directory script and format was created by weasel. The current script uses code written by gsathya & karsten.

This specification was revised after feedback from Damian Johnson (“atagar”) and Iain R. Learmonth (“irl”).

Format Versions

The directory list format uses semantic versioning: https://semver.org

```
   In particular:
     * major versions are used for incompatible changes, like
       removing non-optional fields
     * minor versions are used for compatible changes, like adding
       fields
     * patch versions are for bug fixes, like fixing an
       incorrectly-formatted Summary item

   1.0.0 - The legacy fallback directory list format

   2.0.0 - Adds name and extrainfo structured comments, and section
separator
           comments to make the list easier to parses. Also adds a
source list
           comment to the header.

   3.0.0 - Modifies the format of the source list comment.
```

Future Plans

Tor also has an auth_dirs.inc file, but it is not yet in this format. Tor uses slightly different formats for authorities and fallback directory mirrors, so we will need to make some changes to tor so that it parses this format. (We will also need to add authority-specific information to this format.) See #24818 for details.

We want to add a torrc option so operators can opt-in their relays as fallback directory mirrors. This gives us a signed opt-in confirmation. (We can also continue to accept whitelist entries, and do other checks.) We need to write a short proposal, and make some changes to tor and the fallback update script. See #24839 for details.

Format Details

Directory lists contain the following sections:

- List Header (exactly once)
- List Generation (exactly once, may be empty)
- Directory Entry (zero or more times)

Each section (or entry) ends with a separator.

Nonterminals

The following nonterminals are defined in the Onionoo details document specification:

- dir_address
- fingerprint
- nickname

See https://metrics.torproject.org/onionoo.html#details

The following nonterminals are defined in the “Tor directory protocol” specification in dir-spec.txt:

```
Keyword
ArgumentChar
NL      (newline)
SP      (space)
bool    (must not be confused with Onionoo's JSON "boolean")
```

where arglist is an optional space-separated list of key-value pairs in the form of k=v.

Previously, each line was prepended with the “bridge” keyword, such as

```
"bridge" SP <address:port> NL

"bridge" SP <transportname> SP <address:port> [SP arglist] NL
```

> We don’t do this anymore because Vidalia and TorLauncher don’t expect it. See the commit message for b70347a9c5fd769c6d5d0c0eb5171ace2999a736.

Writing bridge assignments for statistics

BridgeDB can be configured to write bridge assignments to disk for statistical analysis. The start of a bridge assignment is marked by the following line:

“bridge-pool-assignment” SP YYYY-MM-DD HH:MM:SS NL

YYYY-MM-DD HH:MM:SS is the time, in UTC, when BridgeDB has completed loading new bridges and assigning them to distributors.

For every running bridge there is a line with the following format:

fingerprint SP distributor (SP key “=” value)* NL

The distributor is one out of “email”, “https”, or “unallocated”.

Both “email” and “https” distributors support adding keys for “port”, “flag” and “transport”. Respectively, the port number, flag name, and transport types are the values. These are used to indicate that a bridge matches certain port, flag, transport criteria of requests.

The “https” distributor also allows the key “ring” with a number as value to indicate to which IP address area the bridge is returned.

The “unallocated” distributor allows the key “bucket” with the file bucket name as value to indicate which file bucket a bridge is assigned to.

Selecting unallocated bridges to be stored in file buckets

> Kaner should have a look at this section. -NM

```
   BridgeDB can be configured to reserve a subset of bridges and not
give
   them out via one of the distributors.
   BridgeDB assigns reserved bridges to one or more file buckets of
fixed
   sizes and write these file buckets to disk for manual distribution.
   BridgeDB ensures that a file bucket always contains the requested
   number of running bridges.
   If the requested number of bridges in a file bucket is reduced or
the
   file bucket is not required anymore, the unassigned bridges are
   returned to the reserved set of bridges.
   If a bridge stops running, BridgeDB replaces it with another bridge
   from the reserved set of bridges.

> I'm not sure if there's a design bug in file buckets.  What happens
if
> we add a bridge X to file bucket A, and X goes offline?  We would
add
> another bridge Y to file bucket A.  OK, but what if A comes back?
We
> cannot put it back in file bucket A, because it's full.  Are we
going to
> add it to a different file bucket?  Doesn't that mean that most
bridges
> will be contained in most file buckets over time?  -KL
>
> This should be handled the same as if the file bucket is reduced in
size.
> If X returns, then it should be added to the appropriate
distributor. -MF
```

Displaying Bridge Information

After bridges are selected using one of the methods described in Sections 4 - 6, they are output in one of two formats. Bridges are formatted as:

```
<address:port> NL
```

Pluggable transports are formatted as:

```
<transportname> SP <address:port> [SP arglist] NL
```

We derive the following nonterminals from Onionoo and dir-spec.txt:

```
     ipv4_or_port ::= port from an IPv4 or_addresses item

       The ipv4_or_port is the port part of an IPv4 address from the
       Onionoo or_addresses list.

     ipv6_or_address ::= an IPv6 or_addresses item

       The ipv6_or_address is an IPv6 address and port from the
Onionoo
       or_addresses list. The address MAY be in the canonical RFC 5952
       IPv6 address format.

   A key-value pair:

     value ::= Zero or more ArgumentChar, excluding the following
strings:
                 * a double quotation mark (DQUOTE), and
                 * the C comment terminators ("/*" and "*/").

               Note that the C++ comment ("//") and equals sign ("=")
are
               not excluded, because they are reserved for future use
in
               base64 values.

     key_value ::= Keyword "=" value

   We also define these additional nonterminals:

     number ::= An optional negative sign ("-"), followed by one or
more
                numeric characters ([0-9]), with an optional decimal
part
                (".", followed by one or more numeric characters).

     separator ::= "/*" SP+ "=====" SP+ "*/"
```

List Header

The list header consists of a number of key-value pairs, embedded in C-style comments.

List Header Format

```
> requestor will receive new bridges based on rate-limiting and will
(likely)
> eventually work their way around the ring; eventually exhausting all
bridges
> available to them from this distributor. If we use a longer time
period,
> then each time the period expires there will be more bridges in the
ring
> thus reducing the likelihood of all bridges being blocked and
increasing
> the time and effort required to enumerate all bridges. (This is my
> understanding, not from Nick) -MF
>
> Also, we presently need the cache to prevent replays and because if
a user
> sent multiple requests with different criteria in each then we would
leak
> additional bridges otherwise. -MF

   BridgeDB can be configured to include bridge fingerprints in
replies
   along with bridge IP addresses and OR ports.
   BridgeDB can be configured to sign all replies using a PGP signing
key.
   BridgeDB periodically discards old email-address-to-bridge
mappings.
   BridgeDB rejects too frequent email requests coming from the same
   normalized address.
```

To map previously unseen email addresses to a set of bridges, BridgeDB proceeds as follows:

```
     - It normalizes the email address as above, by stripping out
dots,
       removing all of the localpart after the +, and putting it all
       in lowercase.  (Example: "John.Doe+bridges@example.COM" becomes
       "johndoe@example.com".)
     - It maps an HMAC of the normalized address to a position on its
ring
       of bridges.
     - It hands out bridges starting at that position, based on the
       port/flag requirements, as specified at the end of section 4.

    See section 4 for the details of how bridges are selected from the
ring
    and returned to the requestor.
```

```
   BridgeDB can be configured to support one or more distributors that
are
   giving out bridges based on the requestor's email address.
Currently,
   this is how the email distributor works.
   The goal is to bootstrap based on one or more popular email
service's
   sybil prevention algorithms.

> Someone else should look at proposals/ideas/old/xxx-bridge-
disbursement
> to see if this section is missing relevant pieces from it.  -KL

   BridgeDB rejects email addresses containing other characters than
the
   ones that RFC2822 allows.
   BridgeDB may be configured to reject email addresses containing
other
   characters it might not process correctly.

> I don't think we do this, is it worthwhile? -MF

   BridgeDB rejects email addresses coming from other domains than a
   configured set of permitted domains.
   BridgeDB normalizes email addresses by removing "." characters and
by
   removing parts after the first "+" character.
   BridgeDB can be configured to discard requests that do not have the
   value "pass" in their X-DKIM-Authentication-Result header or does
not
   have this header.  The X-DKIM-Authentication-Result header is set
by
   the incoming mail stack that needs to check DKIM authentication.

   BridgeDB does not return a new set of bridges to the same email
address
   until a given time period (typically a few hours) has passed.

> Why don't we fix the bridges we give out for a global 3-hour time
period
> like we do for IP addresses?  This way we could avoid storing email
> addresses.  -KL
>
> The 3-hour value is probably much too short anyway.  If we take
longer
> time values, then people get new bridges when bridges show up, as
> opposed to then we decide to reset the bridges we give them.  (Yes,
this
> problem exists for the IP distributor). -NM
>
> I'm afraid I don't fully understand what you mean here.  Can you
> elaborate?  -KL
>
> Assuming an average churn rate, if we use short time periods, then a
```

```
     "/*" SP+ "type=" Keyword SP+ "*/" SP\* NL

       [At start, exactly once.]

       The type of directory entries in the list. Parsers SHOULD exit
with
       an error if this is not the first line of the list, or if the
value
       is anything other than "fallback".

     "/*" SP+ "version=" version_number SP+ "*/" SP* NL

       [In second position, exactly once.]

       The version of the directory list format.

       version_number is a semantic version, see the "Format Versions"
       section for details.

       Version 1.0.0 represents the undocumented, legacy fallback list
       format(s). Version 2.0.0 and later are documented by this
       specification.

     "/*" SP+ "timestamp=" number SP+ "*/" SP* NL

       [Exactly once.]

       A positive integer that indicates when this directory list was
       generated. This timestamp is guaranteed to increase for every
       version 2.0.0 and later directory list.

       The current timestamp format is YYYYMMDDHHMMSS, as an integer.

     "/*" SP+ "source=" Keyword ("," Keyword)* SP+ "*/" SP* NL

       [Zero or one time.]

       A list of the sources of the directory entries in the list.

       As of version 3.0.0, the possible sources are:
         * "offer-list" - the fallback_offer_list file in the
fallback-scripts
                          repository.
         * "descriptor" - one or more signed descriptors, each
containing an
                          "offer-fallback-dir" line. This feature will
be
                          implemented in ticket #24839.
         * "fallback"   - a fallback_dirs.inc file from a tor
repository.
                          Used in check_existing mode.

       Before #24839 is implemented, the default is "offer-list".
```

```
During the
       transition to signed offers, it will be "descriptor,offer-
list".
       Afterwards, it will be "descriptor".

       In version 2.0.0, only one source name was allowed after
"source=",
       and the deprecated "whitelist" source name was used instead of
       "offer-list".

       This line was added in version 2.0.0 of this specification. The
format
       of this line was modified in version 3.0.0 of this
specification.

     "/*" SP+ key_value SP+ "*/" SP* NL

       [Zero or more times.]

       Future releases may include additional header fields. Parsers
MUST NOT
       rely on the order of these additional fields. Additional header
fields
       will be accompanied by a minor version increment.

     separator SP* NL

       The list header ends with the section separator.
```

List Generation

The list generation information consists of human-readable prose describing the content and origin of this directory list. It is contained in zero or more C-style comments, and may contain multi-line comments and uncommented C code.

In particular, this section may contain C-style comments that contain an equals (“=”) character. It may also be entirely empty.

Future releases may arbitrarily change the content of this section. Parsers MUST NOT rely on a version increment when the format changes.

List Generation Format

In general, parsers MUST NOT rely on the format of this section.

Parsers MAY rely on the following details:

The list generation section MUST NOT be a valid directory entry.

Selecting bridges to be given out based on email addresses

```
      - The first L bridges in the ring after the position that have
the
        port 443, and
      - The first M bridges in the ring after the position that have
the
        flag stable and that it has not already decided to give out,
and
      - The first N-L-M bridges in the ring after the position that it
        has not already decided to give out.

    After BridgeDB selects appropriate bridges to return to the
requestor, it
    then prioritises the ordering of them in a list so that as many
criteria
    are fulfilled as possible within the first few bridges. This list
is then
    truncated to N bridges, if possible. N is currently defined as a
    piecewise function of the number of bridges in the ring such that:

              /
             |  1,   if len(ring) < 20
             |
        N = |  2,   if 20 <= len(ring) <= 100
             |
             |  3,   if 100 <= len(ring)
              \

    The bridges in this sublist, containing no more than N bridges,
are the
    bridges returned to the requestor.
```

The list generation summary MUST end with a section separator:

separator SP* NL

There MUST NOT be any section separators in the list generation section, other than the terminating section separator.

Directory Entry

A directory entry consists of a C string constant, and one or more C-style comments. The C string constant is a valid argument to the DirAuthority or FallbackDir torrc option. The section also contains additional key-value fields in C-style comments.

The list of fallback entries does not include the directory authorities: they are in a separate list. (The Tor implementation combines these lists after parsing them, and applies the DirAuthorityFallbackRate to their weights.)

Directory Entry Format

BridgeDB can be configured to include bridge fingerprints in replies along with bridge IP addresses and OR ports. BridgeDB can be configured to display a CAPTCHA which the user must solve prior to returning the requested bridges.

The current algorithm is as follows. An IP-based distributor splits the bridges uniformly into a set of “rings” based on an HMAC of their ID. Some of these rings are “area” rings for parts of IP space; some are “category” rings for categories of IPs (like proxies). When a client makes a request from an IP, the distributor first sees whether the IP is in one of the categories it knows. If so, the distributor returns an IP from the category rings. If not, the distributor maps the IP into an “area” (that is, a /24), and then uses an HMAC to map the area to one of the area rings.

When the IP-based distributor determines from which area ring it is handing out bridges, it identifies which rules it will use to choose appropriate bridges. Using this information, it searches its cache of rings for one that already adheres to the criteria specified in this request. If one exists, then BridgeDB maps the current “epoch” (N-hour period) and the IP’s area (/24) to a point on the ring based on HMAC, and hands out bridges at that point. If a ring does not already exist which satisfies this request, then a new ring is created and filled with bridges that fulfill the requirements. This ring is then used to select bridges as described.

“Mapping X to Y based on an HMAC” above means one of the following:

```
      - We keep all of the elements of Y in some order, with a mapping
        from all 160-bit strings to positions in Y.
      - We take an HMAC of X using some fixed string as a key to get a
        160-bit value.  We then map that value to the next position of
Y.
```

When giving out bridges based on a position in a ring, BridgeDB first looks at flag requirements and port requirements. For example, BridgeDB may be configured to “Give out at least L bridges with port 443, and at least M bridges with Stable, and at most N bridges total.” To do this, BridgeDB combines to the results:

Selecting bridges to be given out based on IP addresses

```
   BridgeDB may be configured to support one or more distributors
which
   gives out bridges based on the requestor's IP address.  Currently,
this
   is how the HTTPS distributor works.
   The goal is to avoid handing out all the bridges to users in a
similar
   IP space and time.

> Someone else should look at proposals/ideas/old/xxx-bridge-
disbursement
> to see if this section is missing relevant pieces from it.  -KL

   BridgeDB fixes the set of bridges to be returned for a defined time
   period.
   BridgeDB considers all IP addresses coming from the same /24
network
   as the same IP address and returns the same set of bridges. From
here on,
   this non-unique address will be referred to as the IP address's
'area'.
   BridgeDB divides the IP address space equally into a small number
of

> Note, changed term from "areas" to "disjoint clusters" -MF

   disjoint clusters (typically 4) and returns different results for
requests
   coming from addresses that are placed into different clusters.

> I found that BridgeDB is not strict in returning only bridges for a
> given area.  If a ring is empty, it considers the next one.  Is this
> expected behavior?  -KL
>
> This does not appear to be the case, anymore. If a ring is empty,
then
> BridgeDB simply returns an empty set of bridges. -MF
>
> I also found that BridgeDB does not make the assignment to areas
> persistent in the database.  So, if we change the number of rings,
it
> will assign bridges to other rings.  I assume this is okay?  -KL

   BridgeDB maintains a list of proxy IP addresses and returns the
same
   set of bridges to requests coming from these IP addresses.
   The bridges returned to proxy IP addresses do not come from the
same
   set as those for the general IP address space.
```

```
     If a directory entry does not conform to this format, the entry
SHOULD
     be ignored by parsers.

     DQUOTE dir_address SP+ "orport=" ipv4_or_port SP+
       "id=" fingerprint DQUOTE SP* NL

       [At start, exactly once, on a single line.]

       This line consists of the following fields:

       dir_address

         An IPv4 address and DirPort for this directory, as defined by
         Onionoo. In this format version, all IPv4 addresses and
DirPorts
         are guaranteed to be non-zero. (For IPv4 addresses, this
means
         that they are not equal to "0.0.0.0".)

       ipv4_or_port

         An IPv4 ORPort for this directory, derived from Onionoo. In
this
         format version, all IPv4 ORPorts are guaranteed to be non-
zero.

       fingerprint

         The relay fingerprint of this directory, as defined by
Onionoo.
         All relay fingerprints are guaranteed to have one or more
non-zero
         digits.

     Note:

       Each double-quoted C string line that occurs after the first
line,
       starts with space inside the quotes. This is a requirement of
the
       Tor implementation.

     DQUOTE SP+ "ipv6=" ipv6_or_address DQUOTE SP* NL

       [Zero or one time.]

       The IPv6 address and ORPort for this directory, as defined by
       Onionoo. If present, IPv6 addresses and ORPorts are guaranteed
to be
       non-zero. (For IPv6 addresses, this means that they are not
equal to
       "[::]".)
```

```
     DQUOTE SP+ "weight=" number DQUOTE SP* NL

       [Zero or one time.]

       A non-negative, real-numbered weight for this directory.
       The default fallback weight is 1.0, and the default
       DirAuthorityFallbackRate is 1.0 in legacy Tor versions, and 0.1
in
       recent Tor versions.

       weight was removed in version 2.0.0, but is documented because
it
       may be of interest to libraries implementing Tor's fallback
       behaviour.

     DQUOTE SP+ key_value DQUOTE SP* NL

       [Zero or more times.]

       Future releases may include additional data fields in double-
quoted
       C string constants. Parsers MUST NOT rely on the order of these
       additional fields. Additional data fields will be accompanied
by a
       minor version increment.

     "/*" SP+ "nickname=" nickname* SP+ "*/" SP* NL

       [Exactly once.]

       The nickname for this directory, as defined by Onionoo. An
       empty nickname indicates that the nickname is unknown.

       The first fallback list in the 2.0.0 format had nickname lines,
but
       they were all empty.

     "/*" SP+ "extrainfo=" bool SP+ "*/" SP* NL

       [Exactly once.]

       An integer flag that indicates whether this directory caches
       extra-info documents. Set to 1 if the directory claimed that it
       cached extra-info documents in its descriptor when the list was
       created. 0 indicates that it did not, or its descriptor was not
       available.

       The first fallback list in the 2.0.0 format had extrainfo
lines, but
       they were all zero.

     "/*" SP+ key_value SP+ "*/" SP* NL

       [Zero or more times.]
```

(see Section 1). BridgeDB may be configured to give out a different number of bridges (typically 4) depending on the distributor. BridgeDB may define an arbitrary number of rules. These rules may specify the criteria by which a bridge is selected. Specifically, the available rules restrict the IP address version, OR port number, transport type, bridge relay flag, or country in which the bridge should not be blocked.

Parsing extra-info documents

BridgeDB learns if a bridge supports a pluggable transport by parsing extra-info documents. Extra-info documents contain the name of the bridge (but only if it is named), the bridge's fingerprint, the type of pluggable transport(s) it supports, and the IP address and port number on which each transport listens, respectively.

Extra-info documents may contain zero or more entries per bridge. We expect an extra-info entry to contain the following lines in the stated order:

```
"extra-info" SP name SP fingerprint NL
"transport" SP transport SP IP ":" PORT ARGS NL
```

BridgeDB parses the fingerprint, transport type, IP address, port and any arguments that are specified on these lines. BridgeDB skips the name. If the fingerprint is invalid, BridgeDB skips the entry. BridgeDB memorizes the transport type, IP address, port number, and any arguments that are be provided and then it assigns them to the corresponding bridge based on the fingerprint. Arguments are comma-separated and are of the form k=v,k=v. Bridges that do not have an associated extra-info entry are not invalid.

Assigning bridges to distributors

A "distributor" is a mechanism by which bridges are given (or not given) to clients. The current distributors are "email", "https", and "unallocated".

BridgeDB assigns bridges to distributors based on an HMAC hash of the bridge's ID and a secret and makes these assignments persistent. Persistence is achieved by using a database to map node ID to distributor. Each bridge is assigned to exactly one distributor (including the "unallocated" distributor). BridgeDB may be configured to support only a non-empty subset of the distributors specified in this document. BridgeDB may be configured to use different probabilities for assigning new bridges to distributors. BridgeDB does not change existing assignments of bridges to distributors, even if probabilities for assigning bridges to distributors change or distributors are disabled entirely.

Giving out bridges upon requests

Upon receiving a client request, a BridgeDB distributor provides a subset of the bridges assigned to it. BridgeDB only gives out bridges that are contained in the most recently parsed bridge network status and that have the Running flag set

```
       Future releases may include additional data fields in C-style
       comments. Parsers MUST NOT rely on the order of these
additional
       fields. Additional data fields will be accompanied by a minor
version
       increment.

     separator SP* NL

       [Exactly once.]

       Each directory entry ends with the section separator.

     "," SP* NL

       [Exactly once.]

       The comma terminates the C string constant. (Multiple C string
       constants separated by whitespace or comments are coalesced by
       the C compiler.)
```

Usage Considerations

This section contains recommended library behaviours. It does not affect the format of directory lists.

Caching

The fallback list typically changes once every 6-12 months. The data in the list represents the state of the fallback directory entries when the list was created. Fallbacks can and do change their details over time.

Libraries SHOULD parse and cache the most recent version of these lists during their build or release processes. Libraries MUST NOT retrieve the lists by default every time they are deployed or executed.

The latest fallback list can be retrieved from:

https://gitlab.torproject.org/tpo/core/tor/-/raw/main/src/app/config/fallback_dirs.inc?ref_type=heads

Libraries MUST NOT rely on the availability of the server that hosts these lists.

The list can also be retrieved using:

```
git clone https://gitlab.torproject.org/tpo/core/tor/
```

If you just want the latest list, you may wish to perform a shallow clone.

Retrieving Directory Information

Some libraries retrieve directory documents directly from the Tor Directory Authorities. The directory authorities are designed to support Tor relay and client bootstrap, and MAY choose to rate-limit library access. Libraries MAY provide a user-agent in their requests, if they are not intended to support anonymous operation. (User agents are a fingerprinting vector.)

Libraries SHOULD consider the potential load on the authorities, and whether other sources can meet their needs.

Libraries that require high-uptime availability of Tor directory information should investigate the following options:

```
* OnionOO: https://metrics.torproject.org/onionoo.html
  * Third-party OnionOO mirrors are also available
* CollecTor: https://collector.torproject.org/
* Fallback Directory Mirrors
```

Onionoo and CollecTor are typically updated every hour on a regular schedule. Fallbacks update their own directory information at random intervals, see dir-spec for details.

Fallback Reliability

The fallback list is typically regenerated when the fallback failure rate exceeds 25%. Libraries SHOULD NOT rely on any particular fallback being available, or some proportion of fallbacks being available.

Libraries that use fallbacks MAY wish to query an authority after a few fallback queries fail. For example, Tor clients try 3-4 fallbacks before trying an authority.

Sample Data

A sample version 2.0.0 fallback list is available here:

https://trac.torproject.org/projects/tor/raw-attachment/ticket/22759/fallback_dirs_new_format_version.4.inc

A sample transitional version 2.0.0 fallback list is available here:

BridgeDB parses the identity and the publication timestamp from the “r” line, the OR address(es) and ORPort(s) from the “a” line(s), and the assigned flags from the “s” line, specifically checking the assignment of the “Running” and “Stable” flags. BridgeDB memorizes all bridges that have the Running flag as the set of running bridges that can be given out to bridge users. BridgeDB memorizes assigned flags if it wants to ensure that sets of bridges given out should contain at least a given number of bridges with these flags.

Parsing bridge descriptors

BridgeDB learns about a bridge’s most recent IP address and OR port from parsing bridge descriptors. In theory, both IP address and OR port of a bridge are also contained in the “r” line of the bridge network status, so there is no mandatory reason for parsing bridge descriptors. But the functionality described in this section is still implemented in case we need data from the bridge descriptor in the future.

Bridge descriptor files may contain one or more bridge descriptors. We expect a bridge descriptor to contain at least the following lines in the stated order:

```
"@purpose" SP purpose NL
"router" SP nickname SP IP SP ORPort SP SOCKSPort SP DirPort NL
"published" SP timestamp
["opt" SP] "fingerprint" SP fingerprint NL
"router-signature" NL Signature NL
```

BridgeDB parses the purpose, IP, ORPort, nickname, and fingerprint from these lines. BridgeDB skips bridge descriptors if the fingerprint is not contained in the bridge network status parsed earlier or if the bridge does not have the Running flag. BridgeDB discards bridge descriptors which have a different purpose than “bridge”. BridgeDB can be configured to only accept descriptors with another purpose or not discard descriptors based on purpose at all. BridgeDB memorizes the IP addresses and OR ports of the remaining bridges. If there is more than one bridge descriptor with the same fingerprint, BridgeDB memorizes the IP address and OR port of the most recently parsed bridge descriptor. If BridgeDB does not find a bridge descriptor for a bridge contained in the bridge network status parsed before, it does not add that bridge to the set of bridges to be given out to bridge users.

BridgeDB specification

> This document is historical; BridgeDB has been retired, and replaced by rdsys.

This document specifies how BridgeDB processes bridge descriptor files to learn about new bridges, maintains persistent assignments of bridges to distributors, and decides which bridges to give out upon user requests.

Some of the decisions here may be suboptimal: this document is meant to specify current behavior as of August 2013, not to specify ideal behavior.

Importing bridge network statuses and bridge descriptors

BridgeDB learns about bridges by parsing bridge network statuses, bridge descriptors, and extra info documents as specified in Tor's directory protocol. BridgeDB parses one bridge network status file first and at least one bridge descriptor file and potentially one extra info file afterwards.

BridgeDB scans its files on sighup.

BridgeDB does not validate signatures on descriptors or networkstatus files: the operator needs to make sure that these documents have come from a Tor instance that did the validation for us.

Parsing bridge network statuses

Bridge network status documents contain the information of which bridges are known to the bridge authority and which flags the bridge authority assigns to them. We expect bridge network statuses to contain at least the following two lines for every bridge in the given order (format fully specified in Tor's directory protocol):

```
      "r" SP nickname SP identity SP digest SP publication SP IP SP
ORPort
          SP DirPort NL
      "a" SP address ":" port NL (no more than 8 instances)
      "s" SP Flags NL
```

https://raw.githubusercontent.com/teor2345/tor/fallback-format-2-v4/src/or/fallback_dirs.inc

Sample Fallback List Header

```
/*type=fallback */
/* version=2.0.0 */
/* =====*/
```

Sample Fallback List Generation

```
/*Whitelist & blacklist excluded 1326 of 1513 candidates. */
/* Checked IPv4 DirPorts served a consensus within 15.0s. */
/*
Final Count: 151 (Eligible 187, Target 392 (1963* 0.20), Max 200)
Excluded: 36 (Same Operator 27, Failed/Skipped Download 9, Excess 0)
Bandwidth Range: 1.3 - 40.0 MByte/s
*/
/*
Onionoo Source: details Date: 2017-05-16 07:00:00 Version: 4.0
URL: https:onionoo.torproject.orgdetails?
fields=fingerprint%2Cnickname%2Ccontact%2Clast_changed_address_or_port
%2Cconsensus_weight%2Cadvertised_bandwidth%2Cor_addresses%2Cdir_addres
s%2Crecommended_version%2Cflags%2Ceffective_family%2Cplatform&flag=V2D
ir&type=relay&last_seen_days=-0&first_seen_days=30-
*/
/*
Onionoo Source: uptime Date: 2017-05-16 07:00:00 Version: 4.0
URL: https:onionoo.torproject.orguptime?first_seen_days=30-
&flag=V2Dir&type=relay&last_seen_days=-0
*/
/* ===== \*/
```

Sample Fallback Entries

```
"176.10.104.240:80 orport=443
id=0111BA9B604669E636FFD5B503F382A4B7AD6E80"
/*nickname=foo */
/* extrainfo=1 */
/* ===== */
,
"5.9.110.236:9030 orport=9001
id=0756B7CD4DFC8182BE23143FAC0642F515182CEB"
" ipv6=\[2a01:4f8:162:51e2::2\]:9001"
/* nickname= */
/* extrainfo=0 */
/* =====*/
,
```

Historical special hostnames

.noconnect

SYNTAX: [string].noconnect

When Tor sees an address in this format, it immediately closes the connection without attaching it to any circuit. This is useful for controllers that want to test whether a given application is indeed using the same instance of Tor that they're controlling.

This feature was added in Tor 0.1.2.4-alpha, and taken out in Tor 0.2.2.1-alpha over fears that it provided another avenue for detecting Tor users via application-level web tricks.

.onion (version 2)

```
SYNTAX:  [digest].onion
         [ignored].[digest].onion
```

In version 2 .onion addresses, (removed along with the rest of v2 onion services in 0.4.5.11) the "digest" part of the address is the first eighty bits of a SHA-1 digest of the hidden service's identity key (`SHA1(DER(PK))`), encoded in base 32.

The "ignored" portion of the address is intended for use in vhosting, and is supported in Tor 0.2.4.10-alpha and later.

Historical netdoc Items

Historical Server Descriptor Items

These items once appeared in server descriptors

Items are listed in approximate reverse order of withdrawal, oldest last.

`protocols`

```
    "protocols" SP "Link" SP LINK-VERSION-LIST SP "Circuit" SP
           CIRCUIT-VERSION-LIST NL

        [At most once.]
```

An obsolete list of protocol versions, superseded by the "proto" entry. This list was never parsed, and has not been emitted since Tor 0.2.9.4-alpha. New code should neither generate nor parse this line.

`read-history`, `write-history`

```
     "read-history" YYYY-MM-DD HH:MM:SS (NSEC s) NUM,NUM,NUM,NUM,NUM...
NL
         [At most once]
     "write-history" YYYY-MM-DD HH:MM:SS (NSEC s)
NUM,NUM,NUM,NUM,NUM... NL
         [At most once]
```

(These fields once appeared in router descriptors, but have appeared in extra-info descriptors since 0.2.0.x.)

Tor network parameters

This file lists the recognized parameters that can appear on the "params" line of a directory consensus.

Network protocol parameters

"circwindow" – the default package window that circuits should be established with. It started out at 1000 DATA-bearing relay cells, but some research indicates that a lower value would mean fewer cells in transit in the network at any given time. Min: 100, Max: 1000, Default: 1000 First-appeared: Tor 0.2.1.20

"UseOptimisticData" – If set to zero, clients by default shouldn't try to send optimistic data to servers until they have received a CONNECTED message. Min: 0, Max: 1, Default: 1 First-appeared: 0.2.3.3-alpha Default was 0 before: 0.2.9.1-alpha Removed in 0.4.5.1-alpha; now always on.

"usecreatefast" – Used to control whether clients use the CREATE_FAST handshake on the first hop of their circuits. Min: 0, Max: 1. Default: 1. First-appeared: 0.2.4.23, 0.2.5.2-alpha Removed in 0.4.5.1-alpha; now always off.

"min_paths_for_circs_pct" – A percentage threshold that determines whether clients believe they have enough directory information to build circuits. This value applies to the total fraction of bandwidth-weighted paths that the client could build; see path-spec.txt for more information. Min: 25, Max: 95, Default: 60 First-appeared: 0.2.4

"ExtendByEd25519ID" – If true, clients should include Ed25519 identities for relays when generating EXTEND2 messages. Min: 0. Max: 1. Default: 0. First-appeared: 0.3.0

"sendme_emit_min_version" – Minimum SENDME version that can be sent. Min: 0. Max: 255. Default 0. First appeared: 0.4.1.1-alpha.

"sendme_accept_min_version" – Minimum SENDME version that is accepted. Min: 0. Max: 255. Default 0. First appeared: 0.4.1.1-alpha.

"allow-network-reentry" – If true, the Exit relays allow connections that are exiting the network to re-enter. If false, any exit connections going to a relay ORPort or an authority ORPort and DirPort is denied and the stream is terminated. Min: 0. Max: 1. Default: 0 First appeared: 0.4.5.1-alpha.

Performance-tuning parameters

“CircuitPriorityHalflifeMsec” – the halflife parameter used when weighting which circuit will send the next relay cell. Obeyed by Tor 0.2.2.10-alpha and later. (Versions of Tor between 0.2.2.7-alpha and 0.2.2.10-alpha recognized a “CircPriorityHalflifeMsec” parameter, but mishandled it badly.) Min: 1, Max: 2147483647 (INT32_MAX), Default: 30000. First-appeared: Tor 0.2.2.11-alpha

```
    "perconnbwrate" and "perconnbwburst" -- if set, each relay sets up
a
    separate token bucket for every client OR connection, and rate
limits
    that connection independently. Typically left unset, except when
used for
    performance experiments around trac entry 1750. Only honored by
relays
    running Tor 0.2.2.16-alpha and later. (Note that relays running
    0.2.2.7-alpha through 0.2.2.14-alpha looked for bwconnrate and
    bwconnburst, but then did the wrong thing with them; see bug 1830
for
    details.)
    Min: 1, Max: 2147483647 (INT32_MAX), Default: (user setting of
        BandwidthRate/BandwidthBurst).
    First-appeared: 0.2.2.7-alpha
    Removed-in: 0.2.2.16-alpha
```

“NumNTorsPerTAP” – When balancing ntor and TAP requests at relays, how many ntor handshakes should we perform for each TAP handshake? Min: 1. Max: 100000. Default: 10. First-appeared: 0.2.4.17-rc

“circ_max_cell_queue_size” – This parameter determines the maximum number of relay cells allowed per circuit queue. Min: 1000. Max: 2147483647 (INT32_MAX). Default: 50000. First-appeared: 0.3.3.6-rc.

“KISTSchedRunInterval” – How frequently should the “KIST” scheduler run in order to decide which data to write to the network? Value in units of milliseconds. Min: 2. Max: 100. Default: 2 First appeared: 0.3.2

“KISTSchedRunIntervalClient” – How frequently should the “KIST” scheduler run in order to decide which data to write to the network, on clients? Value in units of milliseconds. The client value needs to be much lower than the relay value. Min: 2. Max: 100. Default: 2. First appeared: 0.4.8.2

Historical protocol elements and behaviours

Scope and introduction

This chapter documents historical protocol elements and historical behaviours.

Note: the conventions set out here are not yet followed throughout the Tor specifications.

Definition of “Historical”

A **historical** protocol element or behaviour is one which is neither generated/exhibited, nor recognised/consumed/handled, by any supported version of the Tor implementations.

It should not be necessary to consider historical information to understand the behaviour of the live Tor network. It may be necessary to consider it for some special purposes: for example, as part of developing tools for processing historical data (eg, old consensuses); or, tools which use probing to try to determine the underlying version of possibly mendacious remote software.

If *any* supported Tor implementation generates or handles a behavior, it should be described in the main body, *not* relegated to Historical status. Such a behaviour must be understood to understand the existing live network, and its implementations, even though it may be obsolete or deprecated.

Representation of historical material

Where practical, we move historical material out of the mainline text.

We then document the historical elements here. The structure of this chapter approximately mirrors (as a subset) the structure of the main, current, parts of the spec. Each historical protocol element is placed in its appropriate context.

When that isn’t convenient, historical material remains in the main text. In this case we mark it non-normative, and introduce it with a “Historical” subheading.

Stream: A single application-level connection or request, multiplexed over a Tor circuit. A 'Stream' can currently carry the contents of a TCP connection, a DNS request, or a Tor directory request.

Channel: A pairwise connection between two Tor relays, or between a client and a relay. Circuits are multiplexed over Channels. All channels are currently implemented as TLS connections.

Voting-related parameters

"bwweightscale" – Value that bandwidth-weights are divided by. If not present then this defaults to 10000. Min: 1 First-appeared: 0.2.2.10-alpha

"maxunmeasuredbw" – Used by authorities during voting with method 17 or later. The maximum value to give for any Bandwidth= entry for a router that isn't based on at least three measurements.

(Note: starting in version 0.4.6.1-alpha there was a bug where Tor authorities would instead look at a parameter called "maxunmeasurdbw", without the "e". This bug was fixed in 0.4.9.1-alpha and in 0.4.8.8. Until all relays are running a fixed version, then either this parameter must not be set, or it must be set to the same value for both spellings.)

First-appeared: 0.2.4.11-alpha

"FastFlagMinThreshold", "FastFlagMaxThreshold" – lowest and highest allowable values for the cutoff for routers that should get the Fast flag. This is used during voting to prevent the threshold for getting the Fast flag from being too low or too high. FastFlagMinThreshold: Min: 4. Max: INT32_MAX: Default: 4. FastFlagMaxThreshold: Min: -. Max: INT32_MAX: Default: INT32_MAX First-appeared: 0.2.3.11-alpha

"AuthDirNumSRVAgreements" – Minimum number of agreeing directory authority votes required for a fresh shared random value to be written in the consensus (this rule only applies on the first commit round of the shared randomness protocol). Min: 1. Max: INT32_MAX. Default: 2/3 of the total number of dirauth.

Circuit-build-timeout parameters

"cbtdisabled", "cbtnummodes", "cbtrecentcount", "cbtmaxtimeouts", "cbtmincircs", "cbtquantile", "cbtclosequantile", "cbttestfreq", "cbtmintimeout", "cbtlearntimeout", "cbtmaxopencircs", and "cbtinitialtimeout" – see "2.4.5. Consensus parameters governing behavior" in path-spec.txt for a series of circuit build time related consensus parameters.

Directory-related parameters

“max-consensus-age-to-cache-for-diff” – Determines how much consensus history (in hours) relays should try to cache in order to serve diffs. (min 0, max 8192, default 72)

“try-diff-for-consensus-newer-than” – This parameter determines how old a consensus can be (in hours) before a client should no longer try to find a diff for it. (min 0, max 8192, default 72)

Pathbias parameters

“pb_mincircs”, “pb_noticepct”, “pb_warnpct”, “pb_extremepct”, “pb_dropguards”, “pb_scalecircs”, “pb_scalefactor”, “pb_multfactor”, “pb_minuse”, “pb_noticeusepct”, “pb_extremeusepct”, “pb_scaleuse” – DOCDOC

Family-related parameters

`publish-family-list`: If 1, relays should include any configured or derived family list in their published descriptors. If 0, they should not. (Min: 0, Max: 1. Default: 1.) First-appeared-in: 0.4.9.1-alpha
`use-family-ids`: If 1, clients should consider family IDs (including those from microdescriptors' `family-ids` and those from router descriptors' `family-cert` entries) when building paths. If 0, they should not. (Min: 0, Max: 1. Default: 1) First-appeared-in: 0.4.9.1-alpha
`use-family-lists`: If 1, clients should consider legacy family lists (including `family` entries in router descriptors and microdescriptors) when building paths. If 0, they should not. (Min: 0, Max: 1. Default: 1.) First-appeared-in: 0.4.9.1-alpha

Relay behavior parameters

“refuseunknownexits” – if set to one, exit relays look at the previous hop of circuits that ask to open an exit stream, and refuse to exit if they don’t recognize it as a relay. The goal is to make it harder for people to use them as one-hop proxies. See trac entry 1751 for details. Min: 0, Max: 1 First-appeared: 0.2.2.17-alpha

“onion-key-rotation-days” – (min 1, max 90, default 28)

“onion-key-grace-period-days” – (min 1, max onion-key-rotation-days, default 7)

(“Create”). When we’re talking about a specific cell or message version, we use all-caps (“CREATE”).

For example, a “Create cell” may be CREATE1 or CREATE2, but a “CREATE cell” is never CREATE2.

> This distinction is not perfectly followed everywhere in this spec, and not followed in many older proposals.

Create cell: First part of a handshake, sent by the initiator.

Created cell: Second part of a handshake, sent by the responder.

Extend message: (also known as a RELAY_EXTEND message) First part of a handshake, tunneled through an existing circuit. The last relay in the circuit so far will process this message by decoding it, and sending the appropriate handshake in a Created cell to the client’s chosen next-hop relay.

Extended messge: (also known as a RELAY_EXTENDED message) Second part of a handshake, tunneled through an existing circuit. The last relay in the circuit so far receives the CREATED cell from the new last-hop relay, encodes that cell’s body in in an EXTENDED message, and uses a RELAY cell to deliver the message back to the client. Upon receiving the EXTENDED message, the client’s circuit is one hop longer.

Onion skin: The body of a CREATE/CREATE2 cell or an EXTEND/EXTEND2 message. It contains the first part of the TAP or ntor key establishment handshake.

Hidden Service Protocol

Directory Protocol

General network definitions

Leaky Pipe Topology: The ability for the origin of a circuit to address relay cells to be addressed to any hop in the path of a circuit. In Tor, the destination hop is determined by using the ‘recognized’ field of relay cells.

Messages and cells

Cell: A message sent over a channel. Every cell has an associated command. A cell may be fixed-length or variable-length, depending on its command. Cells are sometimes referred to by their command types: for example, a cell whose command is DESTROY is called a DESTROY cell.

Relay cell: A cell that tells a relay or client about instructions sent over a circuit. The command of a relay cell may be RELAY or RELAY_EARLY. If we need to refer to a cell whose command is specifically RELAY, we call it a "RELAY" cell.

Enveloped relay message: The results of decrypting a relay cell: a relay message plus an associated (optional) StreamID. (If the StreamID is not present, or zero, then the relay message is addressed to the circuit itself rather than to any particular stream on the circuit.)

Relay message: A message sent over a circuit to an individual stream, or to the circuit itself. Relay messages are sometimes referred to by their command types: for example, a message whose command is DATA is sometimes called a DATA message. Sometimes, relay messages are just called "Messages" if no ambiguity would result.

> Note that when prop340 is implemented, the relationship between relay cells and (enveloped) relay messages will no longer be 1:1.

Link handshake

The link handshake establishes the TLS connection over which two Tor participants will send Tor cells. This handshake also authenticates the participants to each other, possibly using Tor cells.

Circuit handshake

Circuit handshakes establish the hop-by-hop onion encryption that clients use to tunnel their application traffic. The client does a pairwise key establishment handshake with each individual relay in the circuit. For every hop except the first, these handshakes tunnel through existing hops in the circuit.

Most of these cell and message types have multiple versions. For example, there is an obsolete CREATE cell, and a modern CREATE2 cell. When we are talking about all versions of a given kind of cell or message, we use an initial capital letter

Every relay should list each onion key it generates for onion-key-rotation-days days after generating it, and then replace it. Relays should continue to accept their most recent previous onion key for an additional onion-key-grace-period-days days after it is replaced. (Introduced in 0.3.1.1-alpha; prior versions of tor hardcoded both of these values to 7 days.)

"AllowNonearlyExtend" – If true, permit EXTEND/EXTEND2 requests that are not inside RELAY_EARLY cells. Min: 0. Max: 1. Default: 0. First-appeared: 0.2.3.11-alpha

"overload_dns_timeout_scale_percent" – This value is a percentage of how many DNS timeout over N seconds we accept before reporting the overload general state. It is scaled by a factor of 1000 in order to be able to represent decimal point. As an example, a value of 1000 means 1%. Min: 0. Max: 100000. Default: 1000. First-appeared: 0.4.6.8 Deprecated: 0.4.7.3-alpha-dev

"overload_dns_timeout_period_secs" – This value is the period in seconds of the DNS timeout measurements (the N in the "overload_dns_timeout_scale_percent" parameter). For this amount of seconds, we will gather DNS statistics and at the end, we'll do an assessment on the overload general signal with regards to DNS timeouts. Min: 0. Max: 2147483647. Default: 600 First-appeared: 0.4.6.8 Deprecated: 0.4.7.3-alpha-dev

"overload_onionskin_ntor_scale_percent" – This value is a percentage of how many onionskin ntor drop over N seconds we accept before reporting the overload general state. It is scaled by a factor of 1000 in order to be able to represent decimal point. As an example, a value of 1000 means 1%. Min: 0. Max: 100000. Default: 1000. First-appeared: 0.4.7.5-alpha

"overload_onionskin_ntor_period_secs" – This value is the period in seconds of the onionskin ntor overload measurements (the N in the "overload_onionskin_ntor_scale_percent" parameter). For this amount of seconds, we will gather onionskin ntor statistics and at the end, we'll do an assessment on the overload general signal. Min: 0. Max: 2147483647. Default: 21600 (6 hours) First-appeared: 0.4.7.5-alpha

"assume-reachable" – If true, relays should publish descriptors even when they cannot make a connection to their IPv4 ORPort. Min: 0. Max: 1. Default: 0. First appeared: 0.4.5.1-alpha.

"assume-reachable-ipv6" – If true, relays should publish descriptors even when they cannot make a connection to their IPv6 ORPort. Min: 0. Max: 1. Default: 0. First appeared: 0.4.5.1-alpha.

“exit_dns_timeout” – The time in milliseconds an Exit sets libevent to wait before it considers the DNS timed out. The corresponding libevent option is “timeout:”. Min: 1. Max: 120000. Default: 1000 (1sec) First appeared: 0.4.7.5-alpha.

“exit_dns_num_attempts” – How many attempts *after the first* should an Exit should try a timing-out DNS query before calling it hopeless? (Each of these attempts will wait for “exit_dns_timeout” independently). The corresponding libevent option is “attempts:”. Min: 0. Max: 255. Default: 2 First appeared: 0.4.7.5-alpha.

"publish-dummy-tap-key" -- If set to 1, relays must include a TAP `onion-key` and `onion-key-crosscert` field in their descriptors, or else authorities may reject those descriptors. Min: 0. Max: 1. Default: 1. First appeared: 0.4.9.x.

V3 onion service parameters

“hs_intro_min_introduce2”, “hs_intro_max_introduce2” – Minimum/maximum amount of INTRODUCE2 messages allowed per circuit before rotation (actual amount picked at random between these two values). Min: 0. Max: INT32_MAX. Defaults: 16384, 32768.

“hs_intro_min_lifetime”, “hs_intro_max_lifetime” – Minimum/maximum lifetime in seconds that a service should keep an intro point for (actual lifetime picked at random between these two values). Min: 0. Max: INT32_MAX. Defaults: 18 hours, 24 hours.

“hs_intro_num_extra” – Number of extra intro points a service is allowed to open. This concept comes from proposal #155. Min: 0. Max: 128. Default: 2.

“hsdir_interval” – The length of a time period, *in minutes*. See rend-spec-v3.txt section [TIME-PERIODS]. Min: 5. Max: 14400. Default: 1440.

“hsdir_n_replicas” – Number of HS descriptor replicas. Min: 1. Max: 16. Default: 2.

“hsdir_spread_fetch” – Total number of HSDirs per replica a tor client should select to try to fetch a descriptor. Min: 1. Max: 128. Default: 3.

“hsdir_spread_store” – Total number of HSDirs per replica a service will upload its descriptor to. Min: 1. Max: 128. Default: 4

“HSV3MaxDescriptorSize” – Maximum descriptor size (in bytes). Min: 1. Max: INT32_MAX. Default: 50000

Hidden Service

A hidden service is a server that will only accept incoming connections via the hidden service protocol. Connection initiators will not be able to learn the IP address of the hidden service, allowing the hidden service to receive incoming connections, serve content, etc, while preserving its location anonymity.

Circuit

An established path through the network, where cryptographic keys are negotiated using the ntor protocol or TAP (Tor Authentication Protocol (deprecated)) with each hop. Circuits can differ in length depending on their purpose. See also Leaky Pipe Topology.

Origin Circuit -

Exit Circuit: A circuit which connects clients to destinations outside the Tor network. For example, if a client wanted to visit duckduckgo.com, this connection would require an exit circuit.

Internal Circuit: A circuit whose traffic never leaves the Tor network. For example, a client could connect to a hidden service via an internal circuit.

Edge connection

```
2.7. Consensus: The state of the Tor network, published every hour,
     decided by a vote from the network's directory authorities.
Clients
     fetch the consensus from directory authorities, fallback
     directories, or directory caches.

2.8. Descriptor: Each descriptor represents information about one
    relay in the Tor network. The descriptor includes the relay's IP
    address, public keys, and other data. Relays send
    descriptors to directory authorities, who vote and publish a
    summary of them in the network consensus.
```

Tor network protocols

Entry relay: The first hop in a Tor circuit. Can be either a guard relay or a bridge, depending on the client's configuration.

Guard relay: A relay that a client uses as its entry for a longer period of time. Guard relays are rotated more slowly to prevent attacks that can come from being exposed to too many guards.

Bridge: A relay intentionally not listed in the public Tor consensus, with the purpose of circumventing entities (such as governments or ISPs) seeking to block clients from using Tor. Currently, bridges are used only as entry relays.

Directory cache: A relay that downloads cached directory information from the directory authorities and serves it to clients on demand. Any relay will act as a directory cache, if its bandwidth is high enough.

Rendezvous point: A relay connecting a client to a hidden service. Each party builds a three-hop circuit, meeting at the rendezvous point.

Client, aka OP (onion proxy)

[Style: the "OP" and "onion proxy" terms are deprecated.]

Authorities

Directory Authority: Nine total in the Tor network, operated by trusted individuals. Directory authorities define and serve the consensus document, defining the "state of the network." This document contains a "router status" section for every relay currently in the network. Directory authorities also serve router descriptors, extra info documents, microdescriptors, and the microdescriptor consensus.

Bridge Authority: One total. Similar in responsibility to directory authorities, but for bridges.

Fallback directory mirror: One of a list of directory caches distributed with the Tor software. (When a client first connects to the network, and has no directory information, it asks a fallback directory. From then on, the client can ask any directory cache that's listed in the directory information it has.)

"hs_service_max_rdv_failures" – This parameter determines the maximum number of rendezvous attempt an HS service can make per introduction. Min 1. Max 10. Default 2. First-appeared: 0.3.3.0-alpha.

"HiddenServiceEnableIntroDoSDefense" – This parameter makes introduction points start using a rate-limiting defense if they support it. Introduction points use this value when no DOS_PARAMS extension is sent in the ESTABLISH_INTRO message. Min: 0. Max: 1. Default: 0. First appeared: 0.4.2.1-alpha.

"HiddenServiceEnableIntroDoSBurstPerSec" – Default maximum burst rate to be used for token bucket for the introduction point rate-limiting. Introduction points use this value when no DOS_PARAMS extension is sent in the ESTABLISH_INTRO message. Min: 0. Max: INT32_MAX. Default: 200 First appeared: 0.4.2.1-alpha.

> Note that the above parameter is slightly misnamed: a burst is not meaningfully "per second".

"HiddenServiceEnableIntroDoSRatePerSec" – Default maximum rate per second to be used for token bucket for the introduction point rate-limiting. Introduction points use this value when no DOS_PARAMS extension is sent. Min: 0. Max: INT32_MAX. Default: 25 First appeared: 0.4.2.1-alpha.

Vanguard parameters

```
    "vanguards-enabled" -- The type of vanguards to use by default
when
    building onion service circuits
        0: No vanguards.
        1: Lite vanguards.
        2: Full vanguards.


    "vanguards-hs-service" -- If higher than vanguards-enabled, and we
are
    running an onion service, we use this level for all our onion
service
    circuits
        0: No vanguards.
        1: Lite vanguards.
        2: Full vanguards.
```

"guard-hs-l2-number" – The number of guards in the L2 guardset Min: 1. Max: INT32_MAX. Default: 4

“guard-hs-l2-lifetime-min” – The minimum lifetime of L2 guards Min: 1. Max: INT32_MAX. Default: 86400 (1 day)

“guard-hs-l2-lifetime-max” – The maximum lifetime of L2 guards Min: 1. Max: INT32_MAX. Default: 1036800 (12 days)

“guard-hs-l3-number” – The number of guards in the L3 guardset Min: 1. Max: INT32_MAX. Default: 8

“guard-hs-l3-lifetime-min” – The minimum lifetime of L3 guards Min: 1. Max: INT32_MAX. Default: 3600 (1 hour)

“guard-hs-l3-lifetime-max” – The maximum lifetime of L3 guards Min: 1. Max: INT32_MAX. Default: 172800 (48 hours)

Denial-of-service parameters

Denial of Service mitigation parameters. Introduced in 0.3.3.2-alpha:

“DoSCircuitCreationEnabled” – Enable the circuit creation DoS mitigation.

“DoSCircuitCreationMinConnections” – Minimum threshold of concurrent connections before a client address can be flagged as executing a circuit creation DoS

“DoSCircuitCreationRate” – Allowed circuit creation rate per second per client IP address once the minimum concurrent connection threshold is reached.

“DoSCircuitCreationBurst” – The allowed circuit creation burst per client IP address once the minimum concurrent connection threshold is reached.

```
"DoSCircuitCreationDefenseType" -- Defense type applied to a
detected client address for the circuit creation mitigation.
    1: No defense.
    2: Refuse circuit creation for the length of
      "DoSCircuitCreationDefenseTimePeriod".
```

“DoSCircuitCreationDefenseTimePeriod” – The base time period that the DoS defense is activated for.

“DoSConnectionEnabled” – Enable the connection DoS mitigation.

“DoSConnectionMaxConcurrentCount” – The maximum threshold of concurrent connection from a client IP address.

Glossary

The Tor Project

This document aims to specify terms, notations, and phrases related to Tor, as used in the Tor specification documents and other documentation.

This glossary is not a design document; it is only a reference.

This glossary is a work-in-progress; double-check its definitions before citing them authoritatively. ;)

Preliminaries

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

Commonly used Tor configuration terms

ORPort - Onion Router Port

DirPort - Directory Port

Tor network components

Relays, aka OR (onion router)

[Style guide: prefer the term “Relay”]

Specific roles

Exit relay: The final hop in an exit circuit before traffic leaves the Tor network to connect to external servers.

Non-exit relay: Relays that send and receive traffic only to other Tor relays and Tor clients.

`expanded@spec.torproject.org` can represent private keys which can be used with the ed25519 signature algorithm, and for which a corresponding working public key is known, but for which there is no known value of `k` (the "real ed25519" private key). Allowing such keys can be convenient when one wishes to find private keys whose public keys have particular patterns, for example when trying to find a `.onion` domain for a Tor Hidden Service. This format is also used where blinded ed25519 keys need to be stored.

```
"DoSConnectionDefenseType" -- Defense type applied to a detected
client address for the connection mitigation. Possible values are:
    1: No defense.
    2: Immediately close new connections.
```

"DoSRefuseSingleHopClientRendezvous" – Refuse establishment of rendezvous points for single hop clients.

"DoSStreamCreationEnabled" – Enable the stream creation DoS mitigation. First appeared: 0.4.9.0-alpha-dev.

"DoSStreamCreationRate" – Allowed stream creation rate per second per circuit. First appeared: 0.4.9.0-alpha-dev.

"DoSStreamCreationBurst" – The allowed stream creation burst per circuit. First appeared: 0.4.9.0-alpha-dev.

```
"DoSStreamCreationDefenseType" -- Defense type applied to a
stream for the stream creation mitigation.
    1: No defense.
    2: Reject the stream or resolve request.
    3: Close the underlying circuit.
First appeared: 0.4.9.0-alpha-dev.
```

Padding-related parameters

"circpad_max_circ_queued_cells" – The circuitpadding module will stop sending more padding relay cells if more than this many cells are in the circuit queue a given circuit. Min: 0. Max: 50000. Default 1000. First appeared: 0.4.0.3-alpha.

"circpad_global_allowed_cells" – This is the number of padding relay cells that must be sent before the 'circpad_global_max_padding_percent' parameter is applied. Min: 0. Max: 65535. Default: 0

"circpad_global_max_padding_pct" – This is the maximum ratio of padding relay cells to total relay cells, specified as a percent. If the global ratio of padding cells to total cells across all circuits exceeds this percent value, no more padding is sent until the ratio becomes lower. 0 means no limit. Min: 0. Max: 100. Default: 0

"circpad_padding_disabled" – If set to 1, no circuit padding machines will negotiate, and all current padding machines will cease padding immediately. Min: 0. Max: 1. Default: 0

“circpad_padding_reduced” – If set to 1, only circuit padding machines marked as “reduced”/“low overhead” will be used. (Currently no such machines are marked as “reduced overhead”). Min: 0. Max: 1. Default: 0

“nf_conntimeout_clients”

- The number of seconds to keep never-used circuits opened and available for clients to use. Note that the actual client timeout is randomized uniformly from this value to twice this value.
- The number of seconds to keep idle (not currently used) canonical channels are open and available. (We do this to ensure a sufficient time duration of padding, which is the ultimate goal.)
- This value is also used to determine how long, after a port has been used, we should attempt to keep building predicted circuits for that port. (See path-spec.txt section 2.1.1.) This behavior was originally added to work around implementation limitations, but it serves as a reasonable default regardless of implementation.
- For all use cases, reduced padding clients use half the consensus value.
- Implementations MAY mark circuits held open past the reduced padding quantity (half the consensus value) as “not to be used for streams”, to prevent their use from becoming a distinguisher. Min: 60. Max: 86400. Default: 1800

“nf_conntimeout_relays” – The number of seconds that idle relay-to-relay connections are kept open. Min: 60. Max: 604800. Default: 3600

“nf_ito_low” – The low end of the range to send padding when inactive, in ms. Min: 0. Max: 60000. Default: 1500

“nf_ito_high” – The high end of the range to send padding, in ms. If nf_ito_low == nf_ito_high == 0, padding will be disabled. Min: nf_ito_low. Max: 60000. Default: 9500

“nf_ito_low_reduced” – For reduced padding clients: the low end of the range to send padding when inactive, in ms. Min: 0. Max: 60000. Default: 9000

“nf_ito_high_reduced” – For reduced padding clients: the high end of the range to send padding, in ms. Min: nf_ito_low_reduced. Max: 60000. Default: 14000

“nf_pad_before_usage” – If set to 1, OR connections are padded before the client uses them for any application traffic. If 0, OR connections are not padded until application data begins. Min: 0. Max: 1. Default: 1

`ed25519-expanded@spec.torproject.org` SHOULD NOT appear in RFC4716 *public* key files. Software which is aware of this key type MUST NOT generate such public key files and SHOULD reject them on loading. (Software handling keys in a type-agnostic manner MAY, and probably will, process such files without complaint.)

> These rules are because public keys should always be advertised as `ed25519` even if the private key is only available as `ed25519-expanded@`: this avoids leaking information about the key generation process to relying parties, and simplifies certification and verification.

> Arti will provide a utility to convert anomalous RFC4716 public key files containing keys declared to be of type `ed25519-expanded@spec.torproject.org` to fully conforming files containg `ed25519` keys. In other circumstances Arti will reject such anomalous files.

The public key data is the same as for the official `ed25519` type:

```
    string         wrapper for the following fixed-length data:
        byte[32]       the actual public key, ENC(A) from RFC8032 3.2
```

(Reference: RFC8032 3.2.)

The private key data is as follows:

```
    string         wrapper for the following fixed-length data:
        byte[32]       ENC(s) as per RFC8032 3.2, "expanded secret
key"
        byte[32]       `h_b...h_(2b-1)` as per RFC8032 3.3,
"separation nonce"
```

(References: `ENC(s)` in RFC8032 3.2; `h_b || ... || h_(2b-1)` as per RFC8032 3.3.)

Keys whose `string` wrapper is not of the expected length MUST be rejected.

> As with `x25519`, the `string` wrapper is useless. We adopt it here for the same reasons.

This private key format does not provide a way to convey or establish a corresponding (unexpanded, standard) ed25519 private key `k`.

> The ed25519 standards define the private key for ed25519 as a 32-byte secret `k`. `k` is then hashed and processed into parameters used for signing, and the public key for verification. The `ed25519-`

The private key data is:

```
string         wrapper for the following fixed-length data:
    byte[32]       the scalar k encoded according to RFC7748 s5
```

k MUST be stored as the true scalar value. So if the private key was generated from 32 random bytes according to the procedure described in RFC7748 s5 "in order to decode 32 random bytes as an integer scalar". the value stored MUST be the "clamped" form: that is, the value *after* the transformation. If a stored value is a byte string which doesn't represent a valid scalar according to RFC7748 s5 (i.e. an "unclamped" value) it SHOULD be rejected; if it is not rejected, it MUST NOT be used unchanged, but MUST instead be clamped.

Keys whose `string` wrapper is not of the expected length MUST be rejected.

> The `string` wrapper is useless, but the same wrapper approach is used in official SSH for ed25519 public keys (RFC8709 s4). and for ed25519 private keys in the SSH agent protocol (draft-miller-ssh-agent-04 4.2.3). We do the same here for consistency (and implementation convenience).

> X25519 keys are interconvertible with ed25519 keys. So, it would be possible to store the ed25519 form instead, and convert on load/save. However, we use a different key type to avoid needing conversions during load/save, and to avoid key type punning and accidental key misuse: using the same key material for different algorithms is a poor idea.

`ed25519-expanded@spec.torproject.org`

These refer to the expanded form of private keys for ed25519 (RFC8032).

This key type appears within OpenSSH private key files. When it does, the `ed25519-expanded@spec.torproject.org` algorithm name is used for the private key (`PROTOCOL.key` section 3, `privatekey1` etc.) but also for the public key (`PROTOCOL.key` section 1, `publickey1` etc.).

> In `PROTOCOL.key` we interpret the requirement that there be "matching" public and private keys to include the requirement that the public key algorithm name strings must be the same.

> In the Arti keystore a private key file whose filename ends with `.ed25519_private` may contain either a standard ed25519 keypair with SSH type `ed25519` or an `ed25519-expanded@spec.torproject.org` keypair.

"nf_pad_relays" – If set to 1, we also pad inactive relay-to-relay connections. Min: 0. Max: 1. Default: 0

"nf_pad_single_onion" – DOCDOC

Guard-related parameters

(See guard-spec.txt for more information on the vocabulary used here.)

"UseGuardFraction" – If true, clients use `GuardFraction` information from the consensus in order to decide how to weight guards when picking them. Min: 0. Max: 1. Default: 0. First appeared: 0.2.6

"guard-lifetime-days" – Controls guard lifetime. If an unconfirmed guard has been sampled more than this many days ago, it should be removed from the guard sample. Min: 1. Max: 3650. Default: 120. First appeared: 0.3.0

"guard-confirmed-min-lifetime-days" – Controls confirmed guard lifetime: if a guard was confirmed more than this many days ago, it should be removed from the guard sample. Min: 1. Max: 3650. Default: 60. First appeared: 0.3.0

"guard-internet-likely-down-interval" – If Tor has been unable to build a circuit for this long (in seconds), assume that the internet connection is down, and treat guard failures as unproven. Min: 1. Max: INT32_MAX. Default: 600. First appeared: 0.3.0

"guard-max-sample-size" – Largest number of guards that clients should try to collect in their sample. Min: 1. Max: INT32_MAX. Default: 60. First appeared: 0.3.0

"guard-max-sample-threshold-percent" – Largest bandwidth-weighted fraction of guards that clients should try to collect in their sample. Min: 1. Max: 100. Default: 20. First appeared: 0.3.0

"guard-meaningful-restriction-percent" – If the client has configured tor to exclude so many guards that the available guard bandwidth is less than this percentage of the total, treat the guard sample as "restricted", and keep it in a separate sample. Min: 1. Max: 100. Default: 20. First appeared: 0.3.0

"guard-extreme-restriction-percent" – Warn the user if they have configured tor to exclude so many guards that the available guard bandwidth is less than this percentage of the total. Min: 1. Max: 100. Default: 1. First appeared: 0.3.0. MAX was INT32_MAX, which would have no meaningful effect. MAX lowered to 100 in 0.4.7.

"guard-min-filtered-sample-size" – If fewer than this number of guards is available in the sample after filtering out unusable guards, the client should try to add more guards to the sample (if allowed). Min: 1. Max: INT32_MAX. Default: 20. First appeared: 0.3.0

"guard-n-primary-guards" – The number of confirmed guards that the client should treat as "primary guards". Min: 1. Max: INT32_MAX. Default: 3. First appeared: 0.3.0

```
    "guard-n-primary-guards-to-use", "guard-n-primary-dir-guards-to-
use"
    -- number of primary guards and primary directory guards that the
    client should be willing to use in parallel.  Other primary guards
    won't get used unless the earlier ones are down.
    "guard-n-primary-guards-to-use":
       Min 1, Max INT32_MAX: Default: 1.
    "guard-n-primary-dir-guards-to-use"
       Min 1, Max INT32_MAX: Default: 3.
    First appeared: 0.3.0
```

"guard-nonprimary-guard-connect-timeout" – When trying to confirm nonprimary guards, if a guard doesn't answer for more than this long in seconds, treat lower-priority guards as usable. Min: 1. Max: INT32_MAX. Default: 15 First appeared: 0.3.0

"guard-nonprimary-guard-idle-timeout" – When trying to confirm nonprimary guards, if a guard doesn't answer for more than this long in seconds, treat it as down. Min: 1. Max: INT32_MAX. Default: 600 First appeared: 0.3.0

"guard-remove-unlisted-guards-after-days" – If a guard has been unlisted in the consensus for at least this many days, remove it from the sample. Min: 1. Max: 3650. Default: 20. First appeared: 0.3.0

Obsolete parameters

"NumDirectoryGuards", "NumEntryGuards" – Number of guard nodes clients should use by default. If NumDirectoryGuards is 0, we default to NumEntryGuards. NumDirectoryGuards: Min: 0. Max: 10. Default: 0 NumEntryGuards: Min: 1. Max: 10. Default: 3 First-appeared: 0.2.4.23, 0.2.5.6-alpha Removed in: 0.3.0

"GuardLifetime" – Duration for which clients should choose guard nodes, in seconds. Min: 30 days. Max: 1826 days. Default: 60 days. First-appeared: 0.2.4.12-alpha Removed in: 0.3.0.

```
    uint32        number of keys, N
         string       publickey1, where the contained binary data is:
              string      public key algorithm name (RFC4250 table
4.11.3)
              byte[]      public key data (algorithm-specific)
         ... keys 2 to N-1 inclusive, each as for publickey1 ...
         string       publickeyN (as for publickey1)
```

Encoding of the private key data

OpenSSH `PROTOCOL.key` defines the representation of the private key data as, simply: "using the same rules as used for SSH agent". However, no specific section is referred to; the SSH agent protocol is only available as an Internet Draft draft-miller-ssh-agent-04; and, the actual encoding used by OpenSSH is hard to relate to that document. So we document our understanding here.

The contents of each `privatekey1`/`privatekey2`/`privatekeyN` is:

```
    string      public key algorithm name (RFC4250 table 4.11.3)
    byte[]      public key data (algorithm-specific)
    byte[]      private key data (algorithm-specific)
```

> Note that this depends on the reader knowing the public key algorithm. The public key data must be self-delimiting, since the file format doesn't provide a separate length field. Although here we discuss only algorithms whose public key data, and private key data, are each a single `string`, that is not always the case: for example, the `ssh-rsa` algorithm's key data formats are sequences of `mpint`s without any surrounding overall length.
>
> Note also that the encrypted section does not separately state the number of keys or their total length. The reader must keep reading until it encounters either the end, or something that looks like padding (starting with a byte 0x01).

`x25519@spec.torproject.org`

These refer to keys for X25519, ie, Diffie-Hellman on Curve25519, as per RFC7748 6.1. and s5.

The public key data is:

```
    string         wrapper for the following fixed-length data:
        byte[32]       the u-coordinate encoded as u[] from RFC7748 s5
```

- In the namespace of public key algorithms - see RFC4250 table 4.11.3, but only when found within an SSH/OpenSSH format key file.
- The meaning of this name is summarised as "X25519 private key"
- The full details can be found at the linked text, which is part of the Arti keystore section, below.

The registered names resemble email addresses, but they are **not email addresses** and mail to them will not be delivered.

For further information, consult the linked specifications.

SSH key types for the Arti keystore

The Arti keystore stores private keys in OpenSSH key format (and, sometimes, public keys, in SSH format). But it needs to store some key types that are not used in the SSH protocols. So the following key types are defined.

These are in the namespace of Public Key Algorithms (RFC4250 4.11.3) but they are only meaningful in OpenSSH format private key files (OpenSSH `PROTOCOL.key` document) and SSH public key files (RFC4716 3.4).

In each case we specify/reference

- the name of the "public key algorithm" (RFC4250 4.11.3),
- the underlying cryptographic algorithm(s),
- the public key data ("key/certificate data" in RFC4253 6.6), see Encoding of the public key data
- the private key data (see Encoding of the private key data)

Encoding of the public key data

OpenSSH `PROTOCOL.key` does not clearly state the contents of the `publickey1`/`publickey2`/`publickeyN` fields in the outer (unencrypted) section (`PROTOCOL.key` s1), so we state it here.

Each `publickey` consists of the encoded public key as per RFC4253 6.6 (under "Certificates and public keys are encoded as follows").

So the overall format of this part of the file is:

"UseNTorHandshake" – If true, then versions of Tor that support NTor will prefer to use it by default. Min: 0, Max: 1. Default: 1. First-appeared: 0.2.4.8-alpha Removed in: 0.2.9.

"Support022HiddenServices" – Used to implement a mass switch-over from sending timestamps to hidden services by default to sending no timestamps at all. If this option is absent, or is set to 1, clients with the default configuration send timestamps; otherwise, they do not. Min: 0, Max: 1. Default: 1. First-appeared: 0.2.4.18-rc Removed in: 0.2.6

Enabling and disabling subprotocol capabilities

Sometimes we want the ability to enable or disable a feature across all relays or clients.

To do so, when the feature corresponds to a subprotocol capability we can follow the following pattern.

Let CAP be the string formed by putting the capability in lowercase and replacing its = with a -. (For example, a hypothetical `Relay=33` capability would be represented with a CAP string of `relay-33`.)

We use CAP to derive some or all of these parameters. All are booleans, and have Min 0, Max 1, and default of 0 or 1.

- `client-CAP`: Tells clients to use the capability where available.
- `relay-CAP`: Tells relays to provide the capability if possible.
- `advertise-CAP`: Tells relays to advertise the capability if they provide it.
- `hsc-CAP`: Tells clients to use the capability with onion services where available.
- `hss-CAP`: Tells onion services to provide the capability if possible.
- `hss-advertise-CAP`: Tells onion services to advertise the capability if they provide it.

> Note that, as a safety measure, implementations must not advertise any capability that they do not provide. The descriptions above give us this property.

These parameters do not automatically exist for every capability. Instead, they are added to the specification when a new capability seems as if it may need to be enabled or disabled in this way.

For rationales, discussion, and example timelines for disabling or enabling features, see proposal 346.

Tor Project SSH protocol extensions

The SSH protocol provides various extension facilities.

The Tor Project has defined some extensions, using the domain-name-based extension facility. The Tor Project uses names ending `@spec.torproject.org`.

Id(s)	Namespace	Summary	Specification link (retrieved at)
`ed25519-expanded@`	Public key algorithm (in SSH/ OpenSSH key file)	Expanded ed25519 private key	Arti keystore types
`x25519@`	Public key algorithm (in SSH/ OpenSSH key file)	X25519 keys	Arti keystore types

Registration process

New entries may be added to this table after peer review by the Tor Project developers, via gitlab merge request.

The specification links may be to external documents, not managed as part of the Tor Specifications. Or, they may be links to specific sections of the Tor Specifications, or to Proposals. External links should be dated, for ease of future reference.

Ideally, before a protocol is deployed, its specification should be transferred to the Tor Specifications (and the link in the table adjusted).

Interpretation

This section uses the notation and conventions from `PROTOCOL.key`) and SSH including RFC4251 s5), not those from the rest of the Tor Specifications.

Interpreting the table

For example, the row for `x25519@` indicates that:

- The Tor Project has assigned `x25519@spec.torproject.org`