Sample data

The following has not been obtained from any real measurement.

Generated by Torflow

This an example version 1.0.0 document:

```
1523911758
node_id=$68A483E05A2ABDCA6DA5A3EF8DB5177638A27F80 bw=760 nick=Test
measured_at=1523911725 updated_at=1523911725 pid_error=4.11374090719
pid_error_sum=4.11374090719 pid_bw=57136645 pid_delta=2.12168374577
circ_fail=0.2 scanner=/filepath
node_id=$96C15995F30895689291F455587BD94CA427B6FC bw=189 nick=Test2
measured_at=1523911623 updated_at=1523911623 pid_error=3.96703337994
pid_error_sum=3.96703337994 pid_bw=47422125 pid_delta=2.65469736988
circ_fail=0.0 scanner=/filepath
```

Generated by sbws version 0.1.0

```
1523911758
version=1.1.0
software=sbws
software_version=0.1.0
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
====

bw=380 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=189 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
```

```
      NymEpoch = a nonnegative integer
      SessionGroup = an integer

      IsoFields = a comma-separated list of IsoField values

      IsoField =
               "CLIENTADDR" /
               "CLIENTPORT" /
               "DESTADDR" /
               "DESTPORT" /
               the name of a field that is valid for STREAM events
```

The circuit ID designates which circuit this stream is attached to. If the stream is unattached, the circuit ID “0” is given. The target indicates the address which the stream is meant to resolve or connect to; it can be “(Tor_internal)” for a virtual stream created by the Tor program to talk to itself.

```
      Reason = "MISC" / "RESOLVEFAILED" / "CONNECTREFUSED" /
               "EXITPOLICY" / "DESTROY" / "DONE" / "TIMEOUT" /
               "NOROUTE" / "HIBERNATING" / "INTERNAL"/
"RESOURCELIMIT" /
               "CONNRESET" / "TORPROTOCOL" / "NOTDIRECTORY" / "END" /
               "PRIVATE_ADDR"
```

The “REASON” field is provided only for FAILED, CLOSED, and DETACHED events, and only if extended events are enabled (see 3.19). Clients MUST accept reasons not listed above. Reasons are as given in tor-spec.txt, except for:

- `END`
  We received a RELAY_END message from the other side of this stream.
- `PRIVATE_ADDR`
  The client tried to connect to a private address like 127.0.0.1 or 10.0.0.1 over Tor.
- [XXXX document more. -NM]

The “REMOTE_REASON” field is provided only when we receive a RELAY_END message, and only if extended events are enabled. It contains the actual reason given by the remote OR for closing the stream. Clients MUST accept reasons not listed above. Reasons are as listed in tor-spec.txt.

“REMAP” events include a Source if extended events are enabled:

```
      Source = "CACHE" / "EXIT"
```

Clients MUST accept sources not listed above. “CACHE” is given if the Tor client decided to remap the address because of a cached answer, and “EXIT” is given if the remote node we queried gave us the new address as a response.

The “SOURCE_ADDR” field is included with NEW and NEWRESOLVE events if extended events are enabled. It indicates the address and port that requested the connection, and can be (e.g.) used to look up the requesting program.

```
Purpose = "DIR_FETCH" / "DIR_UPLOAD" / "DNS_REQUEST" /
          "USER" /  "DIRPORT_TEST"
```

The “PURPOSE” field is provided only for NEW and NEWRESOLVE events, and only if extended events are enabled (see 3.19). Clients MUST accept purposes not listed above. The purposes above are defined as:

- `DIR_FETCH`
  This stream is generated internally to Tor for fetching directory information.
- `DIR_UPLOAD`
  An internal stream for uploading information to a directory authority.
- `DIRPORT_TEST`
  A stream we’re using to test our own directory port to make sure it’s reachable.
- `DNS_REQUEST`
  A user-initiated DNS request.
- `USER`
  This stream is handling user traffic, OR it’s internal to Tor, but it doesn’t match one of the purposes above.

The “SOCKS_USERNAME” and “SOCKS_PASSWORD” fields indicate the credentials that were used by a SOCKS client to connect to Tor’s SOCKS port and initiate this stream. (Streams for SOCKS clients connected with different usernames and/or passwords are isolated on separate circuits if the IsolateSOCKSAuth flag is active; see Proposal 171.)

The “CLIENT_PROTOCOL” field indicates the protocol that was used by a client to initiate this stream. (Streams for clients connected with different protocols are isolated on separate circuits if the IsolateClientProtocol flag is active.) Controllers MUST tolerate unrecognized client protocols.

The “NYM_EPOCH” field indicates the nym epoch that was active when a client initiated this stream. The epoch increments when the NEWNYM signal is received. (Streams with different nym epochs are isolated on separate circuits.)

[Zero or one time.]

The number of times this relay received `XOFF_RECV` stream events while being measured in the last data_period days.

This KeyValue was added in version 1.6.0 of this specification.

“xoff_sent” Int

[Zero or one time.]

The number of times this relay received `XOFF_SENT` stream events while being measured in the last data_period days.

This KeyValue was added in version 1.6.0 of this specification.

“r_strm” Float

[Zero or one time.]

The stream ratio of this relay calculated as explained in B4.3.

This KeyValue was added in version 1.7.0 of this specification.

“r_strm_filt” Float

[Zero or one time.]

The filtered stream ratio of this relay calculated as explained in B4.3.

This KeyValue was added in version 1.7.0 of this specification.

Torflow

Torflow RelayLines include node_id and bw, and other KeyValue pairs [2].

References:

1. https://gitlab.torproject.org/tpo/network-health/torflow
2. https://gitlab.torproject.org/tpo/network-health/torflow/-/blob/main/NetworkScanners/BwAuthority/README.spec.txt?ref_type=heads#L332 The Torflow specification is outdated, and does not match the current implementation. See section A.1. for the format produced by Torflow.
3. The Tor Directory Protocol
4. How Tor Version Numbers Work In Tor
5. https://semver.org/

The “SESSION_GROUP” field indicates the session group of the listener port that a client used to initiate this stream. By default, the session group is different for each listener port, but this can be overridden for a listener via the “SessionGroup” option in torrc. (Streams with different session groups are isolated on separate circuits.)

The “ISO_FIELDS” field indicates the set of STREAM event fields for which stream isolation is enabled for the listener port that a client used to initiate this stream. The special values “CLIENTADDR”, “CLIENTPORT”, “DESTADDR”, and “DESTPORT”, if their correspondingly named fields are not present, refer to the Address and Port components of the “SOURCE_ADDR” and Target fields.

OR Connection status changed

The syntax is:

```
    "650" SP "ORCONN" SP (LongName / Target) SP ORStatus [ SP
"REASON="
             Reason ] [ SP "NCIRCS=" NumCircuits ] [ SP "ID=" ConnID ]
CRLF

    ORStatus = "NEW" / "LAUNCHED" / "CONNECTED" / "FAILED" / "CLOSED"
```

In Tor versions 0.1.2.2-alpha through 0.2.2.1-alpha with feature VERBOSE_NAMES turned off and before version 0.1.2.2-alpha, OR Connection is as follows:

```
    "650" SP "ORCONN" SP (ServerID / Target) SP ORStatus
          [ SP "REASON=" Reason ] [ SP "NCIRCS=" NumCircuits ] CRLF
```

NEW is for incoming connections, and LAUNCHED is for outgoing connections. CONNECTED means the TLS handshake has finished (in either direction). FAILED means a connection is being closed that hasn’t finished its handshake, and CLOSED is for connections that have handshaked.

A LongName or ServerID is specified unless it’s a NEW connection, in which case we don’t know what server it is yet, so we use Address:Port.

If extended events are enabled (see 3.19), optional reason and circuit counting information is provided for CLOSED and FAILED events.

```
      Reason = "MISC" / "DONE" / "CONNECTREFUSED" /
               "IDENTITY" / "CONNECTRESET" / "TIMEOUT" / "NOROUTE" /
               "IOERROR" / "RESOURCELIMIT" / "PT_MISSING"
```

NumCircuits counts both established and pending circuits.

The ORStatus values are as follows:

- `NEW`
  We have received a new incoming OR connection, and are starting the server-side handshake.
- `LAUNCHED`
  We have launched a new outgoing OR connection, and are starting the client-side handshake.
- `CONNECTED`
  The OR connection has been connected and the handshake is done.
- `FAILED`
  Our attempt to open the OR connection failed.
- `CLOSED` The OR connection closed in an unremarkable way.

The Reason values for closed/failed OR connections are:

- `DONE`
  The OR connection has shut down cleanly.
- `CONNECTREFUSED`
  We got an ECONNREFUSED while connecting to the target OR.
- `IDENTITY`
  We connected to the OR, but found that its identity was not what we expected.
- `CONNECTRESET`
  We got an ECONNRESET or similar IO error from the connection with the OR.
- `TIMEOUT`
  We got an ETIMEOUT or similar IO error from the connection with the OR, or we're closing the connection for being idle for too long.
- `NOROUTE`
  We got an ENOTCONN, ENETUNREACH, ENETDOWN, EHOSTUNREACH, or similar error while connecting to the OR.
- `IOERROR`
  We got some other IO error on our connection to the OR.
- `RESOURCELIMIT`
  We don't have enough operating system resources (file descriptors, buffers, etc.) to connect to the OR.
- `PT_MISSING`
  No pluggable transport was available.
- `MISC`
  The OR connection closed for some other reason.

votes that don't change relay weights too much. It also avoids flapping when the reporting threshold is reached.

This KeyValue was added in version 1.4.0 of this specification.

"unmeasured" bool

[Zero or one time.]

If the value is 1, this relay was not successfully measured and Tor bandwidth authorities MAY NOT vote on this relay. (Current Tor versions do not change their behaviour based on the "unmeasured" key.)

If the value is 0 or the KeyValue is not present, this relay was successfully measured.

Because Tor versions released before April 2019 (see section 1.4. for the full list of versions) ignore "vote=0", generator implementations MUST set "bw=1" for unmeasured relays. Using the minimum bw value makes authorities that do not understand "vote=0" or "unmeasured=1" produce votes that don't change relay weights too much.

This KeyValue was added in version 1.4.0 of this specification.

"vote" bool

[Zero or one time.]

If the value is 0, Tor directory authorities SHOULD ignore the relay's entry in the bandwidth file. They SHOULD vote for the relay the same way they would vote for a relay that is not present in the file.

This MAY be the case when this relay was not successfully measured but it is included in the Bandwidth File, to diagnose why they were not measured.

If the value is 1 or the KeyValue is not present, Tor directory authorities MUST use the relay's bw value in any votes for that relay.

Implementations MUST also set "bw=1" for unmeasured relays. But they MUST NOT change the bw for under_min_report relays. (See the explanations under "unmeasured" and "under_min_report" for more details.)

This KeyValue was added in version 1.4.0 of this specification.

"xoff_recv" Int

“relay_recent_measurements_excluded_old_count” Int

[Zero or one time.]

The number of successful measurements for this relay that are too old (more than data_period days, 5 by default).

Excludes measurements that are already counted in relay_recent_measurements_excluded_near_count.

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This KeyValue was added in version 1.4.0 of this specification.

“relay_recent_measurements_excluded_few_count” Int

[Zero or one time.]

The number of successful measurements for this relay that were ignored because the relay did not have enough successful measurements (fewer than 2, by default).

Excludes measurements that are already counted in relay_recent_measurements_excluded_near_count or relay_recent_measurements_excluded_old_count.

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This KeyValue was added in version 1.4.0 of this specification.

“under_min_report” bool

[Zero or one time.]

If the value is 1, there are not enough eligible relays in the bandwidth file, and Tor bandwidth authorities MAY NOT vote on this relay. (Current Tor versions do not change their behaviour based on the “under_min_report” key.)

If the value is 0 or the KeyValue is not present, there are enough relays in the bandwidth file.

Because Tor versions released before April 2019 (see section 1.4. for the full list of versions) ignore “vote=0”, generator implementations MUST NOT change the bandwidths for under_min_report relays. Using the same bw value makes authorities that do not understand “vote=0” or “under_min_report=1” produce

[First added ID parameter in 0.2.5.2-alpha]

Bandwidth used in the last second

The syntax is:

```
      "650" SP "BW" SP BytesRead SP BytesWritten *(SP Type "=" Num)
 CRLF
      BytesRead = 1*DIGIT
      BytesWritten = 1*DIGIT
      Type = "DIR" / "OR" / "EXIT" / "APP" / ...
      Num = 1*DIGIT
```

BytesRead and BytesWritten are the totals.
[In a future Tor version, we may also include a breakdown of the connection types that used bandwidth this second (not implemented yet).]

Log messages

The syntax is:

```
     "650" SP Severity SP ReplyText CRLF
     or
     "650+" Severity CRLF Data 650 SP "OK" CRLF

     Severity = "DEBUG" / "INFO" / "NOTICE" / "WARN"/ "ERR"
```

Some low-level logs may be sent from signal handlers, so their destination logs must be signal-safe. These low-level logs include backtraces, logging function errors, and errors in code called by logging functions. Signal-safe logs are never sent as control port log events.

Control port message trace debug logs are never sent as control port log events, to avoid modifying control output when debugging.

New descriptors available

This event is generated when new router descriptors (not microdescs or extrainfos or anything else) are received.

Syntax:

```
      "650" SP "NEWDESC" 1*(SP LongName) CRLF
```

In Tor versions 0.1.2.2-alpha through 0.2.2.1-alpha with feature VERBOSE_NAMES turned off and before version 0.1.2.2-alpha, it is as follows:

```
    "650" SP "NEWDESC" 1*(SP ServerID) CRLF
```

New Address mapping

These events are generated when a new address mapping is entered in Tor's address map cache, or when the answer for a RESOLVE command is found. Entries can be created by a successful or failed DNS lookup, a successful or failed connection attempt, a RESOLVE command, a MAPADDRESS command, the AutomapHostsOnResolve feature, or the TrackHostExits feature.

Syntax:

```
     "650" SP "ADDRMAP" SP Address SP NewAddress SP Expiry
       [SP "error=" ErrorCode] [SP "EXPIRES=" UTCExpiry] [SP "CACHED="
Cached]
       [SP "STREAMID=" StreamId] CRLF

     NewAddress = Address / "<error>"
     Expiry = DQUOTE ISOTime DQUOTE / "NEVER"

     ErrorCode = "yes" / "internal" / "Unable to launch resolve
request"
     UTCExpiry = DQUOTE IsoTime DQUOTE

     Cached = DQUOTE "YES" DQUOTE / DQUOTE "NO" DQUOTE
     StreamId = DQUOTE StreamId DQUOTE
```

Error and UTCExpiry are only provided if extended events are enabled. The values for Error are mostly useless. Future values will be chosen to match `1*(ALNUM / "_")`; the "Unable to launch resolve request" value is a bug in Tor before 0.2.4.7-alpha.

Expiry is expressed as the local time (rather than UTC). This is a bug, left in for backward compatibility; new code should look at UTCExpiry instead. (If Expiry is "NEVER", UTCExpiry is omitted.)

Cached indicates whether the mapping will be stored until it expires, or if it is just a notification in response to a RESOLVE command.

StreamId is the global stream identifier of the stream or circuit from which the address was resolved.

[Zero or one time.]

The number of times this relay has been prioritized to be measured in the last data_period days.

This KeyValue was added in version 1.4.0 of this specification.

"relay_recent_measurement_attempt_count" Int

[Zero or one time.]

The number of times this relay was tried to be measured in the last data_period days.

This KeyValue was added in version 1.4.0 of this specification.

"relay_recent_measurement_failure_count" Int

[Zero or one time.]

The number of times this relay was tried to be measured in the last data_period days, but it was not possible to obtain a measurement.

This KeyValue was added in version 1.4.0 of this specification.

"relay_recent_measurements_excluded_error_count" Int

[Zero or one time.]

The number of recent relay measurement attempts that failed. Measurements are recent if they are in the last data_period days (5 by default).

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This KeyValue was added in version 1.4.0 of this specification.

"relay_recent_measurements_excluded_near_count" Int

[Zero or one time.]

When all of a relay's recent successful measurements were performed in a period of time that was too short (by default 1 day), the relay is excluded. This KeyValue contains the number of recent successful measurements for the relay that were ignored for this reason.

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This KeyValue was added in version 1.4.0 of this specification.

[Zero or one time.]

The last descriptor observed bandwidth for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“desc_bw_obs_mean” Int

[Zero or one time.]

The descriptor observed bandwidth mean for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“desc_bw_bur” Int

[Zero or one time.]

The descriptor burst bandwidth for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“consensus_bandwidth” Int

[Zero or one time.]

The consensus bandwidth for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“consensus_bandwidth_is_unmeasured” Bool

[Zero or one time.]

If the consensus bandwidth for this relay was not obtained from three or more bandwidth authorities, this KeyValue is True or False otherwise.

This KeyValue was added in version 1.2.0 of this specification.

“relay_in_recent_consensus_count” Int

[Zero or one time.]

The number of times this relay was found in a consensus in the last data_period days. (Unless otherwise stated, data_period is 5 by default.)

This KeyValue was added in version 1.4.0 of this specification.

“relay_recent_priority_list_count” Int

Descriptors uploaded to us in our role as authoritative dirserver

[NOTE: This feature was removed in Tor 0.3.2.1-alpha.]

Tor generates this event when it’s a directory authority, and somebody has just uploaded a server descriptor.

Syntax:

```
"650" "+" "AUTHDIR_NEWDESCS" CRLF Action CRLF Message CRLF
  Descriptor CRLF "." CRLF "650" SP "OK" CRLF
Action = "ACCEPTED" / "DROPPED" / "REJECTED"
Message = Text
```

The Descriptor field is the text of the server descriptor; the Action field is “ACCEPTED” if we’re accepting the descriptor as the new best valid descriptor for its router, “REJECTED” if we aren’t taking the descriptor and we’re complaining to the uploading relay about it, and “DROPPED” if we decide to drop the descriptor without complaining. The Message field is a human-readable string explaining why we chose the Action. (It doesn’t contain newlines.)

Our descriptor changed

Syntax:

```
"650" SP "DESCCHANGED" CRLF
```

[First added in 0.1.2.2-alpha.]

Status events

Status events (STATUS_GENERAL, STATUS_CLIENT, and STATUS_SERVER) are sent based on occurrences in the Tor process pertaining to the general state of the program. Generally, they correspond to log messages of severity Notice or higher. They differ from log messages in that their format is a specified interface.

Syntax:

```
"650" SP StatusType SP StatusSeverity SP StatusAction
                                    [SP StatusArguments] CRLF

StatusType = "STATUS_GENERAL" / "STATUS_CLIENT" / "STATUS_SERVER"
StatusSeverity = "NOTICE" / "WARN" / "ERR"
StatusAction = 1*ALPHA
StatusArguments = StatusArgument *(SP StatusArgument)
StatusArgument = StatusKeyword '=' StatusValue
StatusKeyword = 1*(ALNUM / "_")
StatusValue = 1*(ALNUM / '_')  / QuotedString
```

StatusAction is a string, and StatusArguments is a series of keyword=value pairs on the same line. Values may be space-terminated strings, or quoted strings.

These events are always produced with EXTENDED_EVENTS and VERBOSE_NAMES; see the explanations in the USEFEATURE section for details.

Controllers MUST tolerate unrecognized actions, MUST tolerate unrecognized arguments, MUST tolerate missing arguments, and MUST tolerate arguments that arrive in any order.

Each event description below is accompanied by a recommendation for controllers. These recommendations are suggestions only; no controller is required to implement them.

Compatibility note: versions of Tor before 0.2.0.22-rc incorrectly generated “STATUS_SERVER” as “STATUS_SEVER”. To be compatible with those versions, tools should accept both.

Actions for STATUS_GENERAL events can be as follows:

- `CLOCK_JUMPED`
  . `TIME=NUM`
  Tor spent enough time without CPU cycles that it has closed all its circuits and will establish them anew. This typically happens when a laptop goes to sleep and then wakes up again. It also happens when the system is swapping so heavily that Tor is starving. The “time” argument specifies the number of seconds Tor thinks it was unconscious for (or alternatively, the number of seconds it went back in time).
  This status event is sent as NOTICE severity normally, but WARN severity if Tor is acting as a server currently.
  {Recommendation for controller: ignore it, since we don’t really know what the user should do anyway. Hm.}

The number of times that the bandwidth measurements for this relay failed because the destination Web server was not available.

This KeyValue was added in version 1.4.0 of this specification.

“error_second_relay” Int

[Zero or one time.]

The number of times that the bandwidth measurements for this relay failed because sbws could not find a second relay for the test circuit.

This KeyValue was added in version 1.4.0 of this specification.

“error_misc” Int

[Zero or one time.]

The number of times that the bandwidth measurements for this relay failed because of other reasons.

This KeyValue was added in version 1.1.0 of this specification.

“bw_mean” Int

[Zero or one time.]

The measured bandwidth mean for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“bw_median” Int

[Zero or one time.]

The measured bandwidth median for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“desc_bw_avg” Int

[Zero or one time.]

The descriptor average bandwidth for this relay in bytes per second.

This KeyValue was added in version 1.2.0 of this specification.

“desc_bw_obs_last” Int

[Zero or one time.]

The Round Trip Time in milliseconds to obtain 1 byte of data.

This KeyValue was added in version 1.1.0 of this specification. It became optional in version 1.3.0 or 1.4.0 of this specification.

“time” DateTime

[Exactly once.]

The date and time timestamp in ISO 8601 format and UTC time zone when the last bandwidth was obtained.

This KeyValue was added in version 1.1.0 of this specification. The Torflow equivalent is “measured_at”.

“success” Int

[Zero or one time.]

The number of times that the bandwidth measurements for this relay were successful.

This KeyValue was added in version 1.1.0 of this specification.

“error_circ” Int

[Zero or one time.]

The number of times that the bandwidth measurements for this relay failed because of circuit failures.

This KeyValue was added in version 1.1.0 of this specification. The Torflow equivalent is “circ_fail”.

“error_stream” Int

[Zero or one time.]

The number of times that the bandwidth measurements for this relay failed because of stream failures.

This KeyValue was added in version 1.1.0 of this specification.

“error_destination” Int

[Zero or one time.]

- `DANGEROUS_VERSION`
  . `CURRENT=version`
  . `REASON=NEW/OBSOLETE/UNRECOMMENDED`
  . `RECOMMENDED="version, version, ..."`
  Tor has found that directory servers don’t recommend its version of the Tor software. RECOMMENDED is a comma-and-space-separated string of Tor versions that are recommended. REASON is NEW if this version of Tor is newer than any recommended version, OBSOLETE if this version of Tor is older than any recommended version, and UNRECOMMENDED if some recommended versions of Tor are newer and some are older than this version. (The “OBSOLETE” reason was called “OLD” from Tor 0.1.2.3-alpha up to and including 0.2.0.12-alpha.)
  {Controllers may want to suggest that the user upgrade OLD or UNRECOMMENDED versions. NEW versions may be known-insecure, or may simply be development versions.}

- `TOO_MANY_CONNECTIONS`
  . `CURRENT=NUM` Tor has reached its ulimit -n or whatever the native limit is on file descriptors or sockets. CURRENT is the number of sockets Tor currently has open. The user should really do something about this. The “current” argument shows the number of connections currently open.
  {Controllers may recommend that the user increase the limit, or increase it for them. Recommendations should be phrased in an OS-appropriate way and automated when possible.}

- `BUG`
  . `REASON=STRING` Tor has encountered a situation that its developers never expected, and the developers would like to learn that it happened. Perhaps the controller can explain this to the user and encourage her to file a bug report?
  {Controllers should log bugs, but shouldn’t annoy the user in case a bug appears frequently.}

- `CLOCK_SKEW`
  . `SKEW="+" / "-" SECONDS`
  . `MIN_SKEW="+" / "-" SECONDS`
  .
  `SOURCE="DIRSERV:" IP ":" Port / "NETWORKSTATUS:" IP ":" Port / "OR:" IP ":" Port / "CONSENSUS"`
  If “SKEW” is present, it’s an estimate of how far we are from the time declared in the source. (In other words, if we’re an hour in the past, the value is -3600.) “MIN_SKEW” is present, it’s a lower bound. If the source is a

DIRSERV, we got the current time from a connection to a dirserver. If the source is a NETWORKSTATUS, we decided we're skewed because we got a v2 networkstatus from far in the future. If the source is OR, the skew comes from a NETINFO cell from a connection to another relay. If the source is CONSENSUS, we decided we're skewed because we got a networkstatus consensus from the future.
{Tor should send this message to controllers when it thinks the skew is so high that it will interfere with proper Tor operation. Controllers shouldn't blindly adjust the clock, since the more accurate source of skew info (DIRSERV) is currently unauthenticated.}

- `BAD_LIBEVENT`
  . `METHOD=libevent method`
  . `VERSION=libevent version`
  . `BADNESS="BROKEN" / "BUGGY" / "SLOW"`
  . `RECOVERED="NO" / "YES"`
  Tor knows about bugs in using the configured event method in this version of libevent. "BROKEN" libevents won't work at all; "BUGGY" libevents might work okay; "SLOW" libevents will work fine, but not quickly. If "RECOVERED" is YES, Tor managed to switch to a more reliable (but probably slower!) libevent method.
  {Controllers may want to warn the user if this event occurs, though generally it's the fault of whoever built the Tor binary and there's not much the user can do besides upgrade libevent or upgrade the binary.}

- `DIR_ALL_UNREACHABLE`
  Tor believes that none of the known directory servers are reachable – this is most likely because the local network is down or otherwise not working, and might help to explain for the user why Tor appears to be broken.
  {Controllers may want to warn the user if this event occurs; further action is generally not possible.}

Actions for STATUS_CLIENT events can be as follows:

- `BOOTSTRAP`
  . `PROGRESS=num`
  . `TAG=Keyword`
  . `SUMMARY=String`
  . [`WARNING=String`]
  . [`REASON=Keyword`]
  . [`COUNT=num`]
  . [`RECOMMENDATION=Keyword`]

Implementation details

Writing bandwidth files atomically

To avoid inconsistent reads, implementations SHOULD write bandwidth files atomically. If the file is transferred from another host, it SHOULD be written to a temporary path, then renamed to the V3BandwidthsFile path.

sbws versions 0.7.0 and later write the bandwidth file to an archival location, create a temporary symlink to that location, then atomically rename the symlink to the configured V3BandwidthsFile path.

Torflow does not write bandwidth files atomically.

Additional KeyValue pair definitions

KeyValue pairs in RelayLines that current implementations generate.

Simple Bandwidth Scanner

sbws RelayLines contain these keys:

"node_id" hexdigest

As above.

"bw" Bandwidth

As above.

"nick" nickname

[Exactly once.]

The relay nickname.

Torflow also has a "nick" KeyValue.

"rtt" Int

Torflow also generates consensus weights based on the ratio between the measured bandwidth and the minimum of all descriptor bandwidths (at the time of the measurement). So when an operator reduces the MaxAdvertisedBandwidth for a relay, Torflow reduces that relay's measured bandwidth.

KeyValue

[Zero or more times.]

Future format versions may include additional KeyValue pairs on a RelayLine. Additional KeyValue pairs will be accompanied by a minor version increment.

Implementations MAY add additional relay KeyValue pairs as needed. This specification SHOULD be updated to avoid conflicting meanings for the same Keywords.

Parsers MUST NOT rely on the order of these additional KeyValue pairs.

Additional KeyValue pairs MUST NOT use any keywords specified in the header format. If there are, the parser MAY ignore conflicting keywords.

. [`HOST=QuotedString`]

. [`HOSTADDR=QuotedString`]

Tor has made some progress at establishing a connection to the Tor network, fetching directory information, or making its first circuit; or it has encountered a problem while bootstrapping. This status event is especially useful for users with slow connections or with connectivity problems.

```
 "Progress" gives a number between 0 and 100 for how far through
 the bootstrapping process we are. "Summary" is a string that can
 be displayed to the user to describe the *next* task that Tor
 will tackle, i.e., the task it is working on after sending the
 status event. "Tag" is a string that controllers can use to
 recognize bootstrap phases, if they want to do something smarter
 than just blindly displaying the summary string; see Section 5
 for the current tags that Tor issues.

 The StatusSeverity describes whether this is a normal bootstrap
 phase (severity notice) or an indication of a bootstrapping
 problem (severity warn).

 For bootstrap problems, we include the same progress, tag, and
 summary values as we would for a normal bootstrap event, but we
 also include "warning", "reason", "count", and "recommendation"
 key/value combos. The "count" number tells how many bootstrap
 problems there have been so far at this phase. The "reason"
 string lists one of the reasons allowed in the ORCONN event. The
 "warning" argument string with any hints Tor has to offer about
 why it's having troubles bootstrapping.

 The "reason" values are long-term-stable controller-facing tags
to
 identify particular issues in a bootstrapping step.  The warning
 strings, on the other hand, are human-readable. Controllers
 SHOULD NOT rely on the format of any warning string. Currently
 the possible values for "recommendation" are either "ignore" or
 "warn" -- if ignore, the controller can accumulate the string in
 a pile of problems to show the user if the user asks; if warn,
 the controller should alert the user that Tor is pretty sure
 there's a bootstrapping problem.

 The "host" value is the identity digest (in hex) of the node
we're
 trying to connect to; the "hostaddr" is an address:port
combination,
 where 'address' is an ipv4 or ipv6 address.

 Currently Tor uses recommendation=ignore for the first
 nine bootstrap problem reports for a given phase, and then
```

This KeyValue pair SHOULD be present, see the note under node_id.

This KeyValue was added in version 1.1.0 of this specification.

“bw” Bandwidth

[Exactly once.]

The bandwidth of this relay in kilobytes per second.

No Zero Bandwidths: Tor accepts zero bandwidths, but they trigger bugs in older Tor implementations. Therefore, implementations SHOULD NOT produce zero bandwidths. Instead, they SHOULD use one as their minimum bandwidth. If there are zero bandwidths, the parser MAY ignore them.

Bandwidth Aggregation: Multiple measurements can be aggregated using an averaging scheme, such as a mean, median, or decaying average.

Bandwidth Scaling: Torflow scales bandwidths to kilobytes per second. Other implementations SHOULD use kilobytes per second for their initial bandwidth scaling.

If different implementations or configurations are used in votes for the same network, their measurements MAY need further scaling. See Appendix B for information about scaling, and one possible scaling method.

MaxAdvertisedBandwidth: Bandwidth generators MUST limit the relays’ measured bandwidth based on the MaxAdvertisedBadwidth. A relay’s MaxAdvertisedBandwidth limits the bandwidth-avg in its descriptor. bandwidth-avg is the minimum of MaxAdvertisedBandwidth, BandwidthRate, RelayBandwidthRate, BandwidthBurst, and RelayBandwidthBurst. Therefore, generators MUST limit a relay’s measured bandwidth to its descriptor’s bandwidth-avg. This limit needs to be implemented in the generator, because generators may scale consensus weights before sending them to Tor. Generators SHOULD NOT limit measured bandwidths based on descriptors’ bandwidth-observed, because that penalises new relays.

sbws limits the relay’s measured bandwidth to the bandwidth-avg advertised.

Torflow partitions relays based on their bandwidth. For unmeasured relays, Torflow uses the minimum of all descriptor bandwidths, including bandwidth-avg (MaxAdvertisedBandwidth) and bandwidth-observed. Then Torflow measures the relays in each partition against each other, which implicitly limits a relay’s measured bandwidth to the bandwidths of similar relays.

Relay Line format

It consists of zero or more RelayLines containing relay ids and bandwidths. The relays and their KeyValues are in arbitrary order.

There MUST NOT be multiple KeyValue pairs with the same key in the same RelayLine. If there are, the parser SHOULD choose an arbitrary Value.

There MUST NOT be multiple RelayLines per relay identity (node_id or master_key_ed25519). If there are, parsers SHOULD issue a warning. Parers MAY reject the file, choose an arbitrary RelayLine, or ignore both RelayLines.

If a parser does not recognize any extra material in a RelayLine, the extra material MUST be ignored.

Each RelayLine includes the following KeyValue pairs:

“node_id” hexdigest

[Exactly once.]

The fingerprint for the relay’s RSA identity key.

```
Note: In bandwidth files read by Tor versions earlier than
      0.3.4.1-alpha, node_id MUST NOT be at the end of the Line.
      These authority versions are no longer supported.
```

Current Tor versions ignore master_key_ed25519, so node_id MUST be present in each relay Line.

Implementations of version 1.1.0 and later SHOULD include both node_id and master_key_ed25519. Parsers SHOULD accept Lines that contain at least one of them.

From version 1.9.0 of this specification, “node_id” does not longer need to start with the dollar sign before the hexdigit.

“master_key_ed25519” MasterKey

[Zero or one time.]

The relays’s master Ed25519 key, base64 encoded, without trailing “=“s, to avoid ambiguity with KeyValue “=” character.

```
uses recommendation=warn for subsequent problems at that
phase. Hopefully this is a good balance between tolerating
occasional errors and reporting serious problems quickly.
```

- ENOUGH_DIR_INFO

  Tor now knows enough network-status documents and enough server descriptors that it’s going to start trying to build circuits now. [Newer versions of Tor (0.2.6.2-alpha and later): If the consensus contains Exits (the typical case), Tor will build both exit and internal circuits. If not, Tor will only build internal circuits.]

```
 {Controllers may want to use this event to decide when to
indicate
 progress to their users, but should not interrupt the user's
browsing
 to tell them so.}
```

- NOT_ENOUGH_DIR_INFO

  We discarded expired statuses and server descriptors to fall below the desired threshold of directory information. We won’t try to build any circuits until ENOUGH_DIR_INFO occurs again.

```
 {Controllers may want to use this event to decide when to
indicate
 progress to their users, but should not interrupt the user's
browsing
 to tell them so.}
```

- CIRCUIT_ESTABLISHED

  Tor is able to establish circuits for client use. This event will only be sent if we just built a circuit that changed our mind – that is, prior to this event we didn’t know whether we could establish circuits.

```
 {Suggested use: controllers can notify their users that Tor is
 ready for use as a client once they see this status event. \
[Perhaps
 controllers should also have a timeout if too much time passes
and
 this event hasn't arrived, to give tips on how to troubleshoot.
 On the other hand, hopefully Tor will send further status events
 if it can identify the problem.\]}
```

- `CIRCUIT_NOT_ESTABLISHED`

  `. REASON="EXTERNAL_ADDRESS" /`

  `"DIR_ALL_UNREACHABLE" / "CLOCK_JUMPED"`

  We are no longer confident that we can build circuits. The “reason” keyword provides an explanation: which other status event type caused our lack of confidence.

```
 {Controllers may want to use this event to decide when to
indicate
 progress to their users, but should not interrupt the user's
browsing
 to do so.}
 \[Note: only REASON=CLOCK_JUMPED is implemented currently.\]
```

- `CONSENSUS_ARRIVED`

  Tor has received and validated a new consensus networkstatus. (This event can be delayed a little while after the consensus is received, if Tor needs to fetch certificates.)

- `DANGEROUS_PORT`

  `. PORT=port`

  `. RESULT="REJECT" / "WARN"`

  A stream was initiated to a port that’s commonly used for vulnerable-plaintext protocols. If the Result is “reject”, we refused the connection; whereas if it’s “warn”, we allowed it.

Implementations MAY add additional header Lines as needed. This specification SHOULD be updated to avoid conflicting meanings for the same header keys.

Parsers MUST NOT rely on the order of these additional Lines.

Additional header Lines MUST NOT use any keywords specified in the relay measurements format. If there are, the parser MAY ignore conflicting keywords.

Terminator NL

[Zero or one time.]

The Header List section ends with a Terminator.

In version 1.0.0, Header List ends when the first relay bandwidth is found conforming to the next section.

Implementations of version 1.1.0 and later SHOULD use a 5-character terminator.

Tor 0.4.0.1-alpha and later look for a 5-character terminator, or the first relay bandwidth line. sbws versions 0.1.0 to 1.0.2 used a 4-character terminator, this bug was fixed in 1.0.3.

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This Line was added in version 1.4.0 of this specification.

“tor_version” version_number NL

[Zero or one time.]

The Tor version of the Tor process controlled by the generator.

This Line was added in version 1.4.0 of this specification.

“mu” Int NL

[Zero or one time.]

The network stream bandwidth average calculated as explained in B4.2.

This Line was added in version 1.7.0 of this specification.

“muf” Int NL

[Zero or one time.]

The network stream bandwidth average filtered calculated as explained in B4.2.

This Line was added in version 1.7.0 of this specification.

KeyValue NL

[Zero or more times.]

“dirauth_nickname” NL

[Zero or one time.]

The dirauth’s nickname which publishes this V3BandwidthsFile.

This Line was added in version 1.8.0 of this specification.

There MUST NOT be multiple KeyValue header Lines with the same key. If there are, the parser SHOULD choose an arbitrary Line.

If a parser does not recognize a Keyword in a KeyValue Line, it MUST be ignored.

Future format versions may include additional KeyValue header Lines. Additional header Lines will be accompanied by a minor version increment.

```
 {Controllers should warn their users when this occurs, unless
they
 happen to know that the application using Tor is in fact doing
so
 correctly (e.g., because it is part of a distributed bundle).
They
 might also want some sort of interface to let the user configure
 their RejectPlaintextPorts and WarnPlaintextPorts config
options.}
```

- DANGEROUS_SOCKS
  . PROTOCOL="SOCKS4" / "SOCKS5"
  . ADDRESS=IP ":" port
  A connection was made to Tor’s SOCKS port using one of the SOCKS approaches that doesn’t support hostnames – only raw IP addresses. If the client application got this address from gethostbyname(), it may be leaking target addresses via DNS.

```
 {Controllers should warn their users when this occurs, unless
they
 happen to know that the application using Tor is in fact doing
so
 correctly (e.g., because it is part of a distributed bundle).}
```

- SOCKS_UNKNOWN_PROTOCOL
  . DATA=string
  A connection was made to Tor’s SOCKS port that tried to use it for something other than the SOCKS protocol. Perhaps the user is using Tor as an HTTP proxy? The DATA is the first few characters sent to Tor on the SOCKS port.

```
 {Controllers may want to warn their users when this occurs: it
 indicates a misconfigured application.}
```

- SOCKS_BAD_HOSTNAME
  HOSTNAME=QuotedString
  Some application gave us a funny-looking hostname. Perhaps it is broken? In any case it won’t work with Tor and the user should know.

{Controllers may want to warn their users when this occurs: it usually indicates a misconfigured application.}

Actions for STATUS_SERVER can be as follows:

- EXTERNAL_ADDRESS
  . ADDRESS=IP
  . HOSTNAME=NAME
  . METHOD=CONFIGURED / CONFIGURED_ORPORT / DIRSERV / RESOLVED / INTERFACE / GETHOSTNAME
  Our best idea for our externally visible IP has changed to 'IP'. If 'HOSTNAME' is present, we got the new IP by resolving 'NAME'. If the method is 'CONFIGURED', the IP was given verbatim as the Address configuration option. If the method is 'CONFIGURED_ORPORT', the IP was given verbatim in the ORPort configuration option. If the method is 'RESOLVED', we resolved the Address configuration option to get the IP. If the method is 'GETHOSTNAME', we resolved our hostname to get the IP. If the method is 'INTERFACE', we got the address of one of our network interfaces to get the IP. If the method is 'DIRSERV', a directory server told us a guess for what our IP might be.

  {Controllers may want to record this info and display it to the user.}

- CHECKING_REACHABILITY
  . ORADDRESS=IP:port
  . DIRADDRESS=IP:port
  We're going to start testing the reachability of our external OR port or directory port.

  {This event could affect the controller's idea of server status, but the controller should not interrupt the user to tell them so.}

- REACHABILITY_SUCCEEDED
  . ORADDRESS=IP:port
  . DIRADDRESS=IP:port
  We successfully verified the reachability of our external OR port or directory port (depending on which of ORADDRESS or DIRADDRESS is given.)

[Zero or one time.]

The number of relays that have some successful measurements in the last data_period days (5 by default), but all those measurements were performed in a period of time that was too short (by default 1 day).

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This Line was added in version 1.4.0 of this specification.

"recent_measurements_excluded_old_count" Int NL

[Zero or one time.]

The number of relays that have some successful measurements, but all those measurements are too old (more than 5 days, by default).

Excludes relays that are already counted in recent_measurements_excluded_near_count.

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This Line was added in version 1.4.0 of this specification.

"recent_measurements_excluded_few_count" Int NL

[Zero or one time.]

The number of relays that don't have enough recent successful measurements. (Fewer than 2 measurements in the last 5 days, by default).

Excludes relays that are already counted in recent_measurements_excluded_near_count and recent_measurements_excluded_old_count.

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This Line was added in version 1.4.0 of this specification.

"time_to_report_half_network" Int NL

[Zero or one time.]

The time in seconds that it would take to report measurements about the half of the network, given the number of eligible relays and the time it took in the last days (5 days, by default).

```
      In 2019, with 7000 relays in the network, the Value of this Key
SHOULD be
      approximately:
          80 * (7000 * 0.05) = 28000
      Being 0.05 (5%) the fraction of relays in a priority list and 80
      the approximate number of priority lists (see
      "recent_priority_list_count").

      This Line was added in version 1.4.0 of this specification.

    "recent_measurement_attempt_count" Int NL

      [Zero or one time.]
```

The number of times that any relay has been queued to be measured in the last data_period days. (data_period is 5 by default.)

In 2019, with 7000 relays in the network, the Value of this Key SHOULD be approximately the same as “recent_priority_relay_count”, assuming that there is one attempt to measure a relay for each relay that has been prioritized unless there are system, network or implementation issues.

This Line was added in version 1.4.0 of this specification and removed in version 1.5.0.

“recent_measurement_failure_count” Int NL

[Zero or one time.]

The number of times that the scanner attempted to measure a relay in the last data_period days (5 by default), but the relay has not been measured because of system, network or implementation issues.

This Line was added in version 1.4.0 of this specification.

“recent_measurements_excluded_error_count” Int NL

[Zero or one time.]

The number of relays that have no successful measurements in the last data_period days (5 by default).

(See the note in section 1.4, version 1.4.0, about excluded relays.)

This Line was added in version 1.4.0 of this specification.

“recent_measurements_excluded_near_count” Int NL

```
  {This event could affect the controller's idea of server status,
but
  the controller should not interrupt the user to tell them so.}
```

- `GOOD_SERVER_DESCRIPTOR` We successfully uploaded our server descriptor to at least one of the directory authorities, with no complaints.

```
  {Originally, the goal of this event was to declare "every
authority
  has accepted the descriptor, so there will be no complaints
  about it." But since some authorities might be offline, it's
  harder to get certainty than we had thought. As such, this event
  is equivalent to ACCEPTED_SERVER_DESCRIPTOR below. Controllers
  should just look at ACCEPTED_SERVER_DESCRIPTOR and should ignore
  this event for now.}
```

- `SERVER_DESCRIPTOR_STATUS`
  `STATUS="LISTED" / "UNLISTED"`
  We just got a new networkstatus consensus, and whether we’re in it or not in it has changed. Specifically, status is “listed” if we’re listed in it but previous to this point we didn’t know we were listed in a consensus; and status is “unlisted” if we thought we should have been listed in it (e.g. we were listed in the last one), but we’re not.

```
  {Moving from listed to unlisted is not necessarily cause for
  alarm. The relay might have failed a few reachability tests,
  or the Internet might have had some routing problems. So this
  feature is mainly to let relay operators know when their relay
  has successfully been listed in the consensus.}
```

  [Not implemented yet. We should do this in 0.2.2.x. -RD]

- `NAMESERVER_STATUS`
  `. NS=addr`
  `. STATUS="UP" / "DOWN"`
  `. ERR=message`
  One of our nameservers has changed status.

{This event could affect the controller's idea of server status, but
the controller should not interrupt the user to tell them so.}

- NAMESERVER_ALL_DOWN
  All of our nameservers have gone down.

  {This is a problem; if it happens often without the nameservers coming up again, the user needs to configure more or better nameservers.}

- DNS_HIJACKED
  Our DNS provider is providing an address when it should be saying "NOTFOUND"; Tor will treat the address as a synonym for "NOTFOUND".

  {This is an annoyance; controllers may want to tell admins that their
  DNS provider is not to be trusted.}

- DNS_USELESS
  Our DNS provider is giving a hijacked address instead of well-known websites; Tor will not try to be an exit node.

  {Controllers could warn the admin if the relay is running as an exit node: the admin needs to configure a good DNS server. Alternatively, this happens a lot in some restrictive environments
  (hotels, universities, coffeeshops) when the user hasn't registered.}

- BAD_SERVER_DESCRIPTOR
  . DIRAUTH=addr:port
  . REASON=string
  A directory authority rejected our descriptor. Possible reasons include malformed descriptors, incorrect keys, highly skewed clocks, and so on.

  {Controllers should warn the admin, and try to cope if they can.}

"recent_consensus_count" Int NL

[Zero or one time.].

The number of the different consensuses seen in the last data_period days. (data_period is 5 by default.)

```
      Assuming that Tor clients fetch a consensus every 1-2 hours,
      and that the data_period is 5 days, the Value of this Key SHOULD
 be
      between:
          data_period * 24 / 2 =  60
          data_period * 24     = 120

      This Line was added in version 1.4.0 of this specification.

    "recent_priority_list_count" Int NL

      [Zero or one time.]
```

The number of times that a list with a subset of relays prioritized to be measured has been created in the last data_period days. (data_period is 5 by default.)

```
      In 2019, with 7000 relays in the network, the Value of this Key
 SHOULD be
      approximately:
          data_period * 24 / 1.5 = 80
      Being 1.5 the approximate number of hours it takes to measure a
      priority list of 7000 * 0.05 (350) relays, when the fraction of
 relays
      in a priority list is the 5% (0.05).

      This Line was added in version 1.4.0 of this specification.

    "recent_priority_relay_count" Int NL

      [Zero or one time.]
```

The number of relays that has been in in the list of relays prioritized to be measured in the last data_period days. (data_period is 5 by default.)

The number of relays in the consensus.

This Line was added in version 1.2.0 of this specification.

“percent_eligible_relays” Int NL

[Zero or one time.]

The number of eligible relays, as a percentage of the number of relays in the consensus.

```
      This line SHOULD be equal to:
          (number_eligible_relays * 100.0) / number_consensus_relays
      to the number of relays in the consensus to include in this
file.

      This Line was added in version 1.2.0 of this specification.

    "minimum_number_eligible_relays" Int NL

      [Zero or one time.]
```

The minimum number of relays that SHOULD be included in the bandwidth file. See minimum_percent_eligible_relays for details.

```
      This line SHOULD be equal to:
          number_consensus_relays * (minimum_percent_eligible_relays /
100.0)

      This Line was added in version 1.2.0 of this specification.

    "scanner_country" CountryCode NL

      [Zero or one time.]

      The country, as in political geolocation, where the generator is
run.

      This Line was added in version 1.2.0 of this specification.

    "destinations_countries" CountryCodeList NL

      [Zero or one time.]
```

The country, as in political geolocation, or countries where the destination Web server(s) are located. The destination Web Servers serve the data that the generator retrieves to measure the bandwidth.

This Line was added in version 1.2.0 of this specification.

- `ACCEPTED_SERVER_DESCRIPTOR`
  `. DIRAUTH=addr:port`
  A single directory authority accepted our descriptor.
  // actually notice

```
   {This event could affect the controller's idea of server status,
  but
   the controller should not interrupt the user to tell them so.}
```

- `REACHABILITY_FAILED`
  `. ORADDRESS=IP:port`
  `. DIRADDRESS=IP:port`
  We failed to connect to our external OR port or directory port successfully.

```
   {This event could affect the controller's idea of server
  status.  The
   controller should warn the admin and suggest reasonable steps to
  take.}
```

- `HIBERNATION_STATUS`
  `. STATUS="AWAKE" / "SOFT" / "HARD"`
  Our bandwidth based accounting status has changed, and we are now relaying traffic/rejecting new connections/hibernating.

```
   {This event could affect the controller's idea of server
  status.  The
   controller MAY inform the admin, though presumably the
  accounting was
   explicitly enabled for a reason.}
```

[This event was added in tor 0.2.9.0-alpha.]

Our set of guard nodes has changed

Syntax:

```
"650" SP "GUARD" SP Type SP Name SP Status ... CRLF
Type = "ENTRY"
Name = ServerSpec
  (Identifies the guard affected)
Status = "NEW" | "UP" | "DOWN" | "BAD" | "GOOD" | "DROPPED"
```

The ENTRY type indicates a guard used for connections to the Tor network.

The Status values are:

```
    "NEW"  -- This node was not previously used as a guard; now we
 have
               picked it as one.
    "DROPPED" -- This node is one we previously picked as a guard; we
               no longer consider it to be a member of our guard list.
    "UP"   -- The guard now seems to be reachable.
    "DOWN" -- The guard now seems to be unreachable.
    "BAD"  -- Because of flags set in the consensus and/or values in
 the
               configuration, this node is now unusable as a guard.
    "BAD_L2" -- This layer2 guard has expired or got removed from the
               consensus. This node is removed from the layer2 guard
 set.
    "GOOD" -- Because of flags set in the consensus and/or values in
 the
               configuration, this node is now usable as a guard.

  Controllers must accept unrecognized types and unrecognized
 statuses.
```

Network status has changed

Syntax:

```
    "650" "+" "NS" CRLF 1*NetworkStatus "." CRLF "650" SP "OK" CRLF
```

The event is used whenever our local view of a relay status changes. This happens when we get a new v3 consensus (in which case the entries we see are a duplicate of what we see in the NEWCONSENSUS event, below), but it also happens when we decide to mark a relay as up or down in our local status, for example based on connection attempts.

[First added in 0.1.2.3-alpha]

Bandwidth used on an application stream

The syntax is:

The date and time timestamp in ISO 8601 format and UTC time zone when the first relay bandwidth was obtained.

This Line was added in version 1.1.0 of this specification.

“latest_bandwidth” DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format and UTC time zone of the most recent generator bandwidth result.

This time MUST be identical to the initial Timestamp line.

This duplicate value is included to make the format easier for people to read.

This Line was added in version 1.1.0 of this specification.

“number_eligible_relays” Int NL

[Zero or one time.]

The number of relays that have enough measurements to be included in the bandwidth file.

This Line was added in version 1.2.0 of this specification.

“minimum_percent_eligible_relays” Int NL

[Zero or one time.]

The percentage of relays in the consensus that SHOULD be included in every generated bandwidth file.

If this threshold is not reached, format versions 1.3.0 and earlier SHOULD NOT contain any relays. (Bandwidth files always include a header.)

Format versions 1.4.0 and later SHOULD include all the relays for diagnostic purposes, even if this threshold is not reached. But these relays SHOULD be marked so that Tor does not vote on them. See section 1.4 for details.

The minimum percentage is 60% in Torflow, so sbws uses 60% as the default.

This Line was added in version 1.2.0 of this specification.

“number_consensus_relays” Int NL

[Zero or one time.]

This Line was added in version 1.1.0 of this specification.

Version 1.0.0 documents do not contain this Line, and the version_number is considered to be “1.0.0”.

“software” Value NL

[Zero or one time.]

The name of the software that created the document.

This Line was added in version 1.1.0 of this specification.

Version 1.0.0 documents do not contain this Line, and the software is considered to be “torflow”.

“software_version” Value NL

[Zero or one time.]

The version of the software that created the document. The version may be a version_number, a git commit, or some other version scheme.

This Line was added in version 1.1.0 of this specification.

“file_created” DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format and UTC time zone when the file was created.

This Line was added in version 1.1.0 of this specification.

“generator_started” DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format and UTC time zone when the generator started.

This Line was added in version 1.1.0 of this specification.

“earliest_bandwidth” DateTime NL

[Zero or one time.]

```
"650" SP "STREAM_BW" SP StreamID SP BytesWritten SP BytesRead SP
          Time CRLF
BytesWritten = 1*DIGIT
BytesRead = 1*DIGIT
Time = ISOTime2Frac
```

BytesWritten and BytesRead are the number of bytes written and read by the application since the last STREAM_BW event on this stream.

Note that from Tor’s perspective, *reading* a byte on a stream means that the application *wrote* the byte. That’s why the order of “written” vs “read” is opposite for stream_bw events compared to bw events.

The Time field is provided only in versions 0.3.2.1-alpha and later. It records when Tor created the bandwidth event.

These events are generated about once per second per stream; no events are generated for streams that have not written or read. These events apply only to streams entering Tor (such as on a SOCKSPort, TransPort, or so on). They are not generated for exiting streams.

Per-country client stats

The syntax is:

```
"650" SP "CLIENTS_SEEN" SP TimeStarted SP CountrySummary SP
IPVersions CRLF
```

We just generated a new summary of which countries we’ve seen clients from recently. The controller could display this for the user, e.g. in their “relay” configuration window, to give them a sense that they are actually being useful.

Currently only bridge relays will receive this event, but once we figure out how to sufficiently aggregate and sanitize the client counts on main relays, we might start sending these events in other cases too.

TimeStarted is a quoted string indicating when the reported summary counts from (in UTCS).

The CountrySummary keyword has as its argument a comma-separated, possibly empty set of `countrycode=count` pairs. For example (without linebreak):

```
650-CLIENTS_SEEN TimeStarted="2008-12-25 23:50:43"
CountrySummary=us=16,de=8,uk=8
```

The IPVersions keyword has as its argument a comma-separated set of “protocol-family=count” pairs. For example:

```
IPVersions=v4=16,v6=40
```

Note that these values are rounded, not exact. The rounding algorithm is specified in the description of “geoip-client-origins” in dir-spec.txt.

New consensus networkstatus has arrived

The syntax is:

```
"650" "+" "NEWCONSENSUS" CRLF 1*NetworkStatus "." CRLF "650" SP
"OK" CRLF
```

A new consensus networkstatus has arrived. We include NS-style lines for every relay in the consensus. NEWCONSENSUS is a separate event from the NS event, because the list here represents every usable relay: so any relay *not* mentioned in this list is implicitly no longer recommended.

[First added in 0.2.1.13-alpha]

New circuit buildtime has been set

The syntax is:

```
"650" SP "BUILDTIMEOUT_SET" SP Type SP "TOTAL_TIMES=" Total SP
   "TIMEOUT_MS=" Timeout SP "XM=" Xm SP "ALPHA=" Alpha SP
   "CUTOFF_QUANTILE=" Quantile SP "TIMEOUT_RATE=" TimeoutRate SP
   "CLOSE_MS=" CloseTimeout SP "CLOSE_RATE=" CloseRate
   CRLF
Type = "COMPUTED" / "RESET" / "SUSPENDED" / "DISCARD" / "RESUME"
Total = Integer count of timeouts stored
Timeout = Integer timeout in milliseconds
Xm = Estimated integer Pareto parameter Xm in milliseconds
Alpha = Estimated floating point Paredo parameter alpha
Quantile = Floating point CDF quantile cutoff point for this
timeout
TimeoutRate = Floating point ratio of circuits that timeout
CloseTimeout = How long to keep measurement circs in milliseconds
CloseRate = Floating point ratio of measurement circuits that are
closed
```

A new circuit build timeout time has been set. If Type is “COMPUTED”, Tor has computed the value based on historical data. If Type is “RESET”, initialization or

Header List format

It consists of a Timestamp line and zero or more HeaderLines.

All the header lines MUST conform to the HeaderLine format, except the first Timestamp line.

The Timestamp line is not a HeaderLine to keep compatibility with the legacy Bandwidth File format.

Some header Lines MUST appear in specific positions, as documented below. All other Lines can appear in any order.

If a parser does not recognize any extra material in a header Line, the Line MUST be ignored.

If a header Line does not conform to this format, the Line SHOULD be ignored by parsers.

It consists of:

Timestamp NL

[At start, exactly once.]

The Unix Epoch time in seconds of the most recent generator bandwidth result.

If the generator implementation has multiple threads or subprocesses which can fail independently, it SHOULD take the most recent timestamp from each thread and use the oldest value. This ensures all the threads continue running.

If there are threads that do not run continuously, they SHOULD be excluded from the timestamp calculation.

If there are no recent results, the generator MUST NOT generate a new file.

It does not follow the KeyValue format for backwards compatibility with version 1.0.0.

“version” version_number NL

[In second position, zero or one time.]

The specification document format version. It uses semantic versioning [5].

Tor versions earlier than 0.3.5.1-alpha require all lines in the file to be 510 characters or less. The previous limit was 254 characters in Tor 0.2.6.2-alpha and earlier. Parsers MAY ignore longer Lines.

Note that directory authorities are only supported on the two most recent stable Tor versions, so we expect that line limits will be removed after Tor 0.4.0 is released in 2019.

drastic network changes have caused Tor to reset the timeout back to the default, to relearn again. If Type is “SUSPENDED”, Tor has detected a loss of network connectivity and has temporarily changed the timeout value to the default until the network recovers. If type is “DISCARD”, Tor has decided to discard timeout values that likely happened while the network was down. If type is “RESUME”, Tor has decided to resume timeout calculation.

The Total value is the count of circuit build times Tor used in computing this value. It is capped internally at the maximum number of build times Tor stores (NCIRCUITS_TO_OBSERVE).

The Timeout itself is provided in milliseconds. Internally, Tor rounds this value to the nearest second before using it.

[First added in 0.2.2.7-alpha]

Signal received

The syntax is:

```
"650" SP "SIGNAL" SP Signal CRLF

Signal = "RELOAD" / "DUMP" / "DEBUG" / "NEWNYM" / "CLEARDNSCACHE"
```

A signal has been received and actions taken by Tor. The meaning of each signal, and the mapping to Unix signals, is as defined in section 3.7. Future versions of Tor MAY generate signals other than those listed here; controllers MUST be able to accept them.

If Tor chose to ignore a signal (such as NEWNYM), this event will not be sent. Note that some options (like ReloadTorrcOnSIGHUP) may affect the semantics of the signals here.

Note that the HALT (SIGTERM) and SHUTDOWN (SIGINT) signals do not currently generate any event.

[First added in 0.2.3.1-alpha]

Configuration changed

The syntax is:

```
    StartReplyLine *(MidReplyLine) EndReplyLine

    StartReplyLine = "650-CONF_CHANGED" CRLF
    MidReplyLine = "650-" KEYWORD ["=" VALUE] CRLF
    EndReplyLine = "650 OK"
```

Tor configuration options have changed (such as via a SETCONF or RELOAD signal). KEYWORD and VALUE specify the configuration option that was changed. Undefined configuration options contain only the KEYWORD.

Circuit status changed slightly

The syntax is:

```
    "650" SP "CIRC_MINOR" SP CircuitID SP CircEvent [SP Path]
          [SP "BUILD_FLAGS=" BuildFlags] [SP "PURPOSE=" Purpose]
          [SP "HS_STATE=" HSState] [SP "REND_QUERY=" HSAddress]
          [SP "TIME_CREATED=" TimeCreated]
          [SP "OLD_PURPOSE=" Purpose [SP "OLD_HS_STATE=" HSState]]
CRLF

    CircEvent =
             "PURPOSE_CHANGED" / ; circuit purpose or HS-related state
changed
             "CANNIBALIZED"      ; circuit cannibalized

  Clients MUST accept circuit events not listed above.
```

The “OLD_PURPOSE” field is provided for both PURPOSE_CHANGED and CANNIBALIZED events. The “OLD_HS_STATE” field is provided whenever the “OLD_PURPOSE” field is provided and is a hidden-service-related purpose.

Other fields are as specified in section 4.1.1 above.

[First added in 0.2.3.11-alpha]

Pluggable transport launched

The syntax is:

Definitions

The following nonterminals are defined in Tor directory protocol sections 1.2., 2.1.1., 2.1.3.:

```
    bool
    Int
    SP (space)
    NL (newline)
    KeywordChar
    ArgumentChar
    nickname
    hexdigest (a '$', followed by 40 hexadecimal characters
      ([A-Fa-f0-9]))

  Nonterminal defined section 2 of version-spec.txt [4]:

    version_number

  We define the following nonterminals:

    Line ::= ArgumentChar* NL
    RelayLine ::= KeyValue (SP KeyValue)* NL
    HeaderLine ::= KeyValue NL
    KeyValue ::= Key "=" Value
    Key ::= (KeywordChar | "_")+
    Value ::= ArgumentCharValue+
    ArgumentCharValue ::= any printing ASCII character except NL and
SP.
    Terminator ::= "=====" or "===="
                   Generators SHOULD use a 5-character terminator.
    Timestamp ::= Int
    Bandwidth ::= Int
    MasterKey ::= a base64-encoded Ed25519 public key, with
                  padding characters omitted.
    DateTime ::= "YYYY-MM-DDTHH:MM:SS", as in ISO 8601
    CountryCode ::= Two capital ASCII letters ([A-Z]{2}), as defined
in
                    ISO 3166-1 alpha-2 plus "ZZ" to denote unknown
country
                    (eg the destination is in a Content Delivery
Network).
    CountryCodeList ::= One or more CountryCode(s) separated by a
comma
                        ([A-Z]{2}(,[A-Z]{2})*).
```

Note that key_value and value are defined in Tor directory protocol with different formats to KeyValue and Value here.

Format details

```
  The Bandwidth File MUST contain the following sections:
  - Header List (exactly once), which is a partially ordered list of
    - Header Lines (one or more times), then
  - Relay Lines (zero or more times), in an arbitrary order.
  If it does not contain these sections, parsers SHOULD ignore the
file.
```

```
    "650" SP "TRANSPORT_LAUNCHED" SP Type SP Name SP TransportAddress
 SP Port
    Type = "server" | "client"

    Name = The name of the pluggable transport
    TransportAddress = An IPv4 or IPv6 address on which the pluggable
                       transport is listening for connections
    Port = The TCP port on which it is listening for connections.
```

A pluggable transport called ‘Name’ of type ‘Type’ was launched successfully and is now listening for connections on ‘Address’:‘Port’.

Bandwidth used on an OR or DIR or EXIT connection

The syntax is:

```
     "650" SP "CONN_BW" SP "ID=" ConnID SP "TYPE=" ConnType
              SP "READ=" BytesRead SP "WRITTEN=" BytesWritten CRLF

     ConnType = "OR" /  ; Carrying traffic within the tor network.
 This can
                          either be our own (client) traffic or
 traffic we're
                          relaying within the network.
                "DIR" / ; Fetching tor descriptor data, or
 transmitting
                          descriptors we're mirroring.
                "EXIT"  ; Carrying traffic between the tor network and
 an
                          external destination.

     BytesRead = 1*DIGIT
     BytesWritten = 1*DIGIT
```

Controllers MUST tolerate unrecognized connection types.

BytesWritten and BytesRead are the number of bytes written and read by Tor since the last CONN_BW event on this connection.

These events are generated about once per second per connection; no events are generated for connections that have not read or written. These events are only generated if TestingTorNetwork is set.

[First added in 0.2.5.2-alpha]

Bandwidth used by all streams attached to a circuit

The syntax is:

```
"650" SP "CIRC_BW" SP "ID=" CircuitID SP "READ=" BytesRead SP
         "WRITTEN=" BytesWritten SP "TIME=" Time SP
         "DELIVERED_READ=" DeliveredBytesRead SP
         "OVERHEAD_READ=" OverheadBytesRead SP
         "DELIVERED_WRITTEN=" DeliveredBytesWritten SP
         "OVERHEAD_WRITTEN=" OverheadBytesWritten SP
         "SS=" SlowStartState SP
         "CWND=" CWNDCells SP
         "RTT=" RTTMilliseconds SP
         "MIN_RTT=" RTTMilliseconds CRLF

BytesRead = 1*DIGIT
BytesWritten = 1*DIGIT
OverheadBytesRead = 1*DIGIT
OverheadBytesWritten = 1*DIGIT
DeliveredBytesRead = 1*DIGIT
DeliveredBytesWritten = 1*DIGIT
SlowStartState = 0 or 1
CWNDCells = 1*DIGIT
RTTMilliseconds= 1*DIGIT
Time = ISOTime2Frac
```

BytesRead and BytesWritten are the number of bytes read and written on this circuit since the last CIRC_BW event. These bytes have not necessarily been validated by Tor, and can include invalid cells, dropped cells, and ignored cells (such as padding cells). These values include the relay headers, but not circuit headers.

Circuit data that has been validated and processed by Tor is further broken down into two categories: delivered relay message bodies, and overhead. DeliveredBytesRead and DeliveredBytesWritten are the total length of the relay message bodies transmitted since the last CIRC_BW event, not counting relay cell headers or circuit headers. OverheadBytesRead and OverheadBytesWritten are the extra unused bytes at the end of each cell in order for it to be the fixed CELL_LEN bytes long.

The sum of DeliveredBytesRead and OverheadBytesRead MUST be less than BytesRead, and the same is true for their written counterparts. This sum represents the total relay cell bytes on the circuit that have been validated by Tor, not counting relay headers and cell headers. Subtracting this sum (plus relay cell headers) from the BytesRead (or BytesWritten) value gives the byte count that Tor has decided to reject due to protocol errors, or has otherwise decided to ignore.

```
  1.1.0 - Adds a header containing information about the bandwidth
          file. Document the sbws and Torflow relay line keys.

  1.2.0 - If there are not enough eligible relays, the bandwidth file
          SHOULD contain a header, but no relays. (To match Torflow's
          existing behaviour.)

          Adds scanner and destination countries to the header.
          Adds new KeyValue Lines to the Header List section with
          statistics about the number of relays included in the file.
          Adds new KeyValues to Relay Bandwidth Lines, with different
          bandwidth values (averages and descriptor bandwidths).

  1.4.0 - Adds monitoring KeyValues to the header and relay lines.

          RelayLines for excluded relays MAY be present in the
bandwidth
          file for diagnostic reasons. Similarly, if there are not
enough
          eligible relays, the bandwidth file MAY contain all known
relays.

          Diagnostic relay lines SHOULD be marked with vote=0, and
          Tor SHOULD NOT use their bandwidths in its votes.

          Also adds Tor version.
  1.5.0 - Removes "recent_measurement_attempt_count" KeyValue.
  1.6.0 - Adds congestion control stream events KeyValues.
  1.7.0 - Adds ratios KeyValues to the relay lines and network
averages
          KeyValues to the header.
  1.8.0 - Adds "dirauth_nickname" KeyValue to the header.
  1.9.0 - Allows "node_id" KeyValue without the dollar sign at the
start of the
          hexdigit.
  All Tor versions can consume format version 1.0.0.
```

All Tor versions can consume format version 1.1.0 and later, but Tor versions earlier than 0.3.5.1-alpha warn if the header contains any KeyValue lines after the Timestamp.

```
  Tor versions 0.4.0.3-alpha, 0.3.5.8, 0.3.4.11, and earlier do not
  understand "vote=0". Instead, they will vote for the actual
bandwidths
  that sbws puts in diagnostic relay lines:
    * 1 for relays with "unmeasured=1", and
    * the relay's measured and scaled bandwidth when
"under_min_report=1".
```

Scope and 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.

Acknowledgements

The original bandwidth generator (Torflow) and format was created by mike. Teor suggested to write this specification while contributing on pastly’s new bandwidth generator implementation.

This specification was revised after feedback from:

Nick Mathewson (nickm) Iain Learmonth (irl)

Outline

The Tor directory protocol (dir-spec.txt [3]) sections 3.4.1 and 3.4.2, use the term bandwidth measurements, to refer to what here is called Bandwidth File.

A Bandwidth File contains information on relays’ bandwidth capacities and is produced by bandwidth generators, previously known as bandwidth scanners.

Format Versions

1.0.0 - The legacy Bandwidth File format

The Time field is provided only in versions 0.3.2.1-alpha and later. It records when Tor created the bandwidth event.

The SS, CWND, RTT, and MIN_RTT fields are present only if the circuit has negotiated congestion control to an onion service or Exit hop (any intermediate leaky pipe congestion control hops are not examined here). SS provides an indication if the circuit is in slow start (1), or not (0). CWND is the size of the congestion window in terms of number of cells. RTT is the N_EWMA smoothed current RTT value, and MIN_RTT is the minimum RTT value of the circuit. The SS and CWND fields apply only to the upstream direction of the circuit. The slow start state and CWND values of the other endpoint may be different.

These events are generated about once per second per circuit; no events are generated for circuits that had no attached stream writing or reading.

[First added in 0.2.5.2-alpha]

[DELIVERED_READ, OVERHEAD_READ, DELIVERED_WRITTEN, and OVERHEAD_WRITTEN were added in Tor 0.3.4.0-alpha]

[SS, CWND, RTT, and MIN_RTT were added in Tor 0.4.7.5-alpha]

Per-circuit cell stats

The syntax is:

```
      "650" SP "CELL_STATS"
               [ SP "ID=" CircuitID ]
               [ SP "InboundQueue=" QueueID SP "InboundConn=" ConnID ]
               [ SP "InboundAdded=" CellsByType ]
               [ SP "InboundRemoved=" CellsByType SP
                    "InboundTime=" MsecByType ]
               [ SP "OutboundQueue=" QueueID SP "OutboundConn="
 ConnID ]
               [ SP "OutboundAdded=" CellsByType ]
               [ SP "OutboundRemoved=" CellsByType SP
                    "OutboundTime=" MsecByType ] CRLF

      CellsByType, MsecByType = CellType ":" 1*DIGIT
                                0*( "," CellType ":" 1*DIGIT )
      CellType = 1*( "a" - "z" / "0" - "9" / "_" )
```

Examples are:

```
650 CELL_STATS ID=14 OutboundQueue=19403 OutboundConn=15
    OutboundAdded=create_fast:1,relay_early:2
    OutboundRemoved=create_fast:1,relay_early:2
    OutboundTime=create_fast:0,relay_early:0

650 CELL_STATS InboundQueue=19403 InboundConn=32
    InboundAdded=relay:1,created_fast:1
    InboundRemoved=relay:1,created_fast:1
    InboundTime=relay:0,created_fast:0
    OutboundQueue=6710 OutboundConn=18
    OutboundAdded=create:1,relay_early:1
    OutboundRemoved=create:1,relay_early:1
    OutboundTime=create:0,relay_early:0
```

ID is the locally unique circuit identifier that is only included if the circuit originates at this node.

Inbound and outbound refer to the direction of relay cell flow through the circuit which is either to origin (inbound) or from origin (outbound).

InboundQueue and OutboundQueue are identifiers of the inbound and outbound circuit queues of this circuit. These identifiers are only unique per OR connection. OutboundQueue is chosen by this node and matches InboundQueue of the next node in the circuit.

InboundConn and OutboundConn are locally unique IDs of inbound and outbound OR connection. OutboundConn does not necessarily match InboundConn of the next node in the circuit.

InboundQueue and InboundConn are not present if the circuit originates at this node. OutboundQueue and OutboundConn are not present if the circuit (currently) ends at this node.

InboundAdded and OutboundAdded are total number of cells by cell type added to inbound and outbound queues. Only present if at least one cell was added to a queue.

InboundRemoved and OutboundRemoved are total number of cells by cell type processed from inbound and outbound queues. InboundTime and OutboundTime are total waiting times in milliseconds of all processed cells by cell type. Only present if at least one cell was removed from a queue.

These events are generated about once per second per circuit; no events are generated for circuits that have not added or processed any cell. These events are only generated if TestingTorNetwork is set.

Tor Bandwidth File Format

juga
teor

This document describes the format of Tor's Bandwidth File, version 1.0.0 and later.

It is a new specification for the existing bandwidth file format, which we call version 1.0.0. It also specifies new format versions 1.1.0 and later, which are backwards compatible with 1.0.0 parsers.

Since Tor version 0.2.4.12-alpha, the directory authorities use the Bandwidth File file called "V3BandwidthsFile" generated by Torflow [1]. The details of this format are described in Torflow's README.spec.txt. We also summarise the format in this specification.

versions should be compared. EXTRA_INFO may appear any number of times. Tools should generally not parse EXTRA_INFO entries.

Now, we start each development branch with (say) 0.1.1.1-alpha. The patchlevel increments consistently as the status tag changes, for example, as in: 0.1.1.2-alpha, 0.1.1.3-alpha, 0.1.1.4-rc, 0.1.1.5-rc. Eventually, we release 0.1.1.6. The next patch release is 0.1.1.7.

Between these releases, CVS is versioned with a -cvs tag: after 0.1.1.1-alpha comes 0.1.1.1-alpha-cvs, and so on. But starting with 0.1.2.1-alpha-dev, we switched to SVN and started using the “-dev” suffix instead of the “-cvs” suffix.

Version status

```
  Sometimes we need to determine whether a Tor version is obsolete,
  experimental, or neither, based on a list of recommended versions.
The
  logic is as follows:

   * If a version is listed on the recommended list, then it is
     "recommended".

   * If a version is newer than every recommended version, that
version
     is "experimental" or "new".

   * If a version is older than every recommended version, it is
     "obsolete" or "old".

   * The first three components (major,minor,micro) of a version
number
     are its "release series".  If a version has other recommended
     versions with the same release series, and the version is newer
     than all such recommended versions, but it is not newer than
     _every_ recommended version, then the version is "new in series".

   * Finally, if none of the above conditions hold, then the version
is
     "un-recommended."
```

[First added in 0.2.5.2-alpha]

Token buckets refilled

The syntax is:

```
      "650" SP "TB_EMPTY" SP BucketName [ SP "ID=" ConnID ] SP
               "READ=" ReadBucketEmpty SP "WRITTEN=" WriteBucketEmpty
 SP
               "LAST=" LastRefill CRLF

      BucketName = "GLOBAL" / "RELAY" / "ORCONN"
      ReadBucketEmpty = 1*DIGIT
      WriteBucketEmpty = 1*DIGIT
      LastRefill = 1*DIGIT
```

Examples are:

```
      650 TB_EMPTY ORCONN ID=16 READ=0 WRITTEN=0 LAST=100
      650 TB_EMPTY GLOBAL READ=93 WRITTEN=93 LAST=100
      650 TB_EMPTY RELAY READ=93 WRITTEN=93 LAST=100
```

This event is generated when refilling a previously empty token bucket. BucketNames “GLOBAL” and “RELAY” keywords are used for the global or relay token buckets, BucketName “ORCONN” is used for the token buckets of an OR connection. Controllers MUST tolerate unrecognized bucket names.

ConnID is only included if the BucketName is “ORCONN”.

If both global and relay buckets and/or the buckets of one or more OR connections run out of tokens at the same time, multiple separate events are generated.

ReadBucketEmpty (WriteBucketEmpty) is the time in millis that the read (write) bucket was empty since the last refill. LastRefill is the time in millis since the last refill.

If a bucket went negative and if refilling tokens didn’t make it go positive again, there will be multiple consecutive TB_EMPTY events for each refill interval during which the bucket contained zero tokens or less. In such a case, ReadBucketEmpty or WriteBucketEmpty are capped at LastRefill in order not to report empty times more than once.

These events are only generated if TestingTorNetwork is set.

[First added in 0.2.5.2-alpha]

HiddenService descriptors

The syntax is:

```
    "650" SP "HS_DESC" SP Action SP HSAddress SP AuthType SP HsDir
             [SP DescriptorID] [SP "REASON=" Reason] [SP "REPLICA="
Replica]
             [SP "HSDIR_INDEX=" HSDirIndex]

    Action =  "REQUESTED" / "UPLOAD" / "RECEIVED" / "UPLOADED" /
"IGNORE" /
              "FAILED" / "CREATED"
    HSAddress = 16*Base32Character / 56*Base32Character / "UNKNOWN"
    AuthType = "NO_AUTH" / "BASIC_AUTH" / "STEALTH_AUTH" / "UNKNOWN"
    HsDir = LongName / Fingerprint / "UNKNOWN"
    DescriptorID = 32*Base32Character / 43*Base64Character
    Reason = "BAD_DESC" / "QUERY_REJECTED" / "UPLOAD_REJECTED" /
"NOT_FOUND" /
             "UNEXPECTED" / "QUERY_NO_HSDIR" / "QUERY_RATE_LIMITED"
    Replica = 1*DIGIT
    HSDirIndex = 64*HEXDIG
```

These events will be triggered when required HiddenService descriptor is not found in the cache and a fetch or upload with the network is performed.

If the fetch was triggered with only a DescriptorID (using the HSFETCH command for instance), the HSAddress only appears in the Action=RECEIVED since there is no way to know the HSAddress from the DescriptorID thus the value will be "UNKNOWN".

If we already had the v0 descriptor, the newly fetched v2 descriptor will be ignored and a "HS_DESC" event with "IGNORE" action will be generated.

For HsDir, LongName is always preferred. If HsDir cannot be found in node list at the time event is sent, Fingerprint will be used instead.

If Action is "FAILED", Tor SHOULD send Reason field as well. Possible values of Reason are:

- "BAD_DESC" - descriptor was retrieved, but found to be unparsable.
- "QUERY_REJECTED" - query was rejected by HS directory.
- "UPLOAD_REJECTED" - descriptor was rejected by HS directory.
- "NOT_FOUND" - HS descriptor with given identifier was not found.
- "UNEXPECTED" - nature of failure is unknown.

How Tor Version Numbers Work

The Old Way

Before 0.1.0, versions were of the format:

MAJOR.MINOR.MICRO(status(PATCHLEVEL))?(-cvs)?

where MAJOR, MINOR, MICRO, and PATCHLEVEL are numbers, status is one of "pre" (for an alpha release), "rc" (for a release candidate), or "." for a release. As a special case, "a.b.c" was equivalent to "a.b.c.0". We compare the elements in order (major, minor, micro, status, patchlevel, cvs), with "cvs" preceding non-cvs.

We would start each development branch with a final version in mind: say, "0.0.8". Our first pre-release would be "0.0.8pre1", followed by (for example) "0.0.8pre2-cvs", "0.0.8pre2", "0.0.8pre3-cvs", "0.0.8rc1", "0.0.8rc2-cvs", and "0.0.8rc2". Finally, we'd release 0.0.8. The stable CVS branch would then be versioned "0.0.8.1-cvs", and any eventual bugfix release would be "0.0.8.1".

The New Way

Starting at 0.1.0.1-rc, versions are of the format:

`MAJOR.MINOR.MICRO[.PATCHLEVEL][-STATUS_TAG][ (EXTRA_INFO)]*`

The stuff in parentheses is optional. As before, MAJOR, MINOR, MICRO, and PATCHLEVEL are numbers, with an absent number equivalent to 0. All versions should be distinguishable purely by those four numbers.

The STATUS_TAG is purely informational, and lets you know how stable we think the release is: "alpha" is pretty unstable; "rc" is a release candidate; and no tag at all means that we have a final release. If the tag ends with "-cvs" or "-dev", you're looking at a development snapshot that came after a given release. If we *do* encounter two versions that differ only by status tag, we compare them lexically. The STATUS_TAG can't contain whitespace.

The EXTRA_INFO is also purely informational, often containing information about the SCM commit this version came from. It is surrounded by parentheses and can't contain whitespace. Unlike the STATUS_TAG this never impacts the way that

```
 [Newer versions of Tor (0.2.6.2-alpha and later):
  If the consensus contains Exits (the typical case), Tor will build
both
  exit and internal circuits. At this stage, Tor will be ready to
handle
  an application requesting an exit circuit to services like the World
  Wide Web.
```

If the consensus does not contain Exits, Tor will only build internal circuits. In this case, earlier statuses will have included “internal” as indicated above. At this stage, Tor will be ready to handle an application requesting an internal circuit to hidden services at “.onion” addresses.

If a future consensus contains Exits, exit circuits may become available.]

- “QUERY_NO_HSDIR” - No suitable HSDir were found for the query.
- “QUERY_RATE_LIMITED” - query for this service is rate-limited

For “QUERY_NO_HSDIR” or “QUERY_RATE_LIMITED”, the HsDir will be set to “UNKNOWN” which was introduced in tor 0.3.1.0-alpha and 0.4.1.0-alpha respectively.

If Action is “CREATED”, Tor SHOULD send Replica field as well. The Replica field contains the replica number of the generated descriptor. The Replica number is specified in rend-spec.txt section 1.3 and determines the descriptor ID of the descriptor.

For hidden service v3, the following applies:

- The “HSDIR_INDEX=” is an optional field that is only for version 3 which contains the computed index of the HsDir the descriptor was uploaded to or fetched from.
- The “DescriptorID” key is the descriptor blinded key used for the index value at the “HsDir”.
- The “REPLICA=” field is not used for the “CREATED” event because v3 doesn’t use the replica number in the descriptor ID computation.
- Because client authentication is not yet implemented, the “AuthType” field is always “NO_AUTH”.

[HS v3 support added 0.3.3.1-alpha]

HiddenService descriptors content

The syntax is:

```
"650" "+" "HS_DESC_CONTENT" SP HSAddress SP DescId SP HsDir CRLF
          Descriptor CRLF "." CRLF "650" SP "OK" CRLF

HSAddress = 16*Base32Character / 56*Base32Character / "UNKNOWN"
DescId = 32*Base32Character / 32*Base64Character
HsDir = LongName / "UNKNOWN"
Descriptor = The text of the descriptor formatted as specified in
             rend-spec.txt section 1.3 (v2) or rend-spec-v3.txt
             section 2.4 (v3) or empty string on failure.
```

This event is triggered when a successfully fetched HS descriptor is received. The text of that descriptor is then replied. If the HS_DESC event is enabled, it is replied just after the RECEIVED action.

If a fetch fails, the Descriptor is an empty string and HSAddress is set to “UNKNOWN”. The HS_DESC event should be used to get more information on the failed request.

If the fetch fails for the QUERY_NO_HSDIR or QUERY_RATE_LIMITED reason from the HS_DESC event, the HsDir is set to “UNKNOWN”. This was introduced in 0.3.1.0-alpha and 0.4.1.0-alpha respectively.

It’s expected to receive a reply relatively fast as in it’s the time it takes to fetch something over the Tor network. This can be between a couple of seconds up to 60 seconds (not a hard limit). But, in any cases, this event will reply either the descriptor’s content or an empty one.

[HS_DESC_CONTENT was added in Tor 0.2.7.1-alpha] [HS v3 support added 0.3.3.1-alpha]

Network liveness has changed

Syntax:

```
"650" SP "NETWORK_LIVENESS" SP Status CRLF
Status = "UP" /  ; The network now seems to be reachable.
         "DOWN" /  ; The network now seems to be unreachable.
```

Controllers MUST tolerate unrecognized status types.

[NETWORK_LIVENESS was added in Tor 0.2.7.2-alpha]

Pluggable Transport Logs

Syntax:

```
 [Newer versions of Tor (0.2.6.2-alpha and later):
  If the consensus contains Exits (the typical case), Tor will build
both
  exit and internal circuits. If not, Tor will only build internal
circuits.
  In this case, this status will include "internal(ly)" as indicated
above.]

  Phase 85:
  tag=handshake_or summary="Finishing handshake with first hop[ of
internal
                            circuit]"
```

This phase is similar to the “handshake_dir” phase, but it gets reached if we finish a TCP connection to a Tor relay and we have already reached the “conn_or” phase. We’ll stay in this phase until we complete a TLS handshake with a Tor relay.

```
 [Newer versions of Tor (0.2.6.2-alpha and later):
  If the consensus contains Exits (the typical case), Tor may be
finishing
  a handshake with the first hop if either an exit or internal
circuit. In
  this case, it won't specify which type. If the consensus contains no
Exits,
  Tor will only build internal circuits. In this case, this status
will
  include "internal" as indicated above.]
```

Phase 90: tag=circuit_create summary=“Establishing a[n internal] Tor circuit”

Once we’ve finished our TLS handshake with the first hop of a circuit, we will set about trying to make some 3-hop circuits in case we need them soon.

```
 [Newer versions of Tor (0.2.6.2-alpha and later):
  If the consensus contains Exits (the typical case), Tor will build
both
  exit and internal circuits. If not, Tor will only build internal
circuits.
  In this case, this status will include "internal" as indicated
above.]
```

Phase 100: tag=done summary=“Done”

A full 3-hop circuit has been established. Tor is ready to handle application connections now.

```
  [Newer versions of Tor (0.2.6.2-alpha and later):
   If the consensus contains Exits (the typical case), Tor will ask for
   descriptors for both exit and internal paths. If not, Tor will only
ask
   for descriptors for internal paths. In this case, this status will
   include "internal" as indicated above.]

   Phase 50:
   tag=loading_descriptors summary="Loading relay descriptors[ for
internal
                                    paths]"
```

We will ask for relay descriptors from several different locations, so this step will probably make up the bulk of the bootstrapping, especially for users with slow connections. We stay in this phase until we have descriptors for a significant fraction of the usable relays listed in the networkstatus consensus (this can be between 25% and 95% depending on Tor's configuration and network consensus parameters). This phase is also a good opportunity to use the "progress" keyword to indicate partial steps.

```
  [Newer versions of Tor (0.2.6.2-alpha and later):
   If the consensus contains Exits (the typical case), Tor will
download
   descriptors for both exit and internal paths. If not, Tor will only
   download descriptors for internal paths. In this case, this status
will
   include "internal" as indicated above.]
```

Phase 80: tag=conn_or summary="Connecting to the Tor network[internally]"

Once we have a valid consensus and enough relay descriptors, we choose entry guard(s) and start trying to build some circuits. This step is similar to the "conn_dir" phase above; the only difference is the context.

If a Tor starts with enough recent cached directory information, its first bootstrap status event will be for the conn_or phase.

```
    "650" SP "PT_LOG" SP PT=Program SP Message

    Program = The program path as defined in the *TransportPlugin
              configuration option. Tor accepts relative and full
path.
    Message = The log message that the PT sends back to the tor parent
              process minus the "LOG" string prefix. Formatted as
              specified in pt-spec.txt section "3.3.4. Pluggable
              Transport Log Message".
```

This event is triggered when tor receives a log message from the PT.

Example:

```
    PT (obfs4): LOG SEVERITY=debug MESSAGE="Connected to bridge A"
```

The resulting control port event would be:

```
    650 PT_LOG PT=/usr/bin/obs4proxy SEVERITY=debug MESSAGE="Connected
to bridge A"
```

[PT_LOG was added in Tor 0.4.0.1-alpha]

Pluggable Transport Status

Syntax:

```
    "650" SP "PT_STATUS" SP PT=Program SP TRANSPORT=Transport SP
Message

    Program = The program path as defined in the *TransportPlugin
              configuration option. Tor accepts relative and full
path.
    Transport = This value indicates a hint on what the PT is such as
the
                name or the protocol used for instance.
    Message = The status message that the PT sends back to the tor
parent
              process minus the "STATUS" string prefix. Formatted as
              specified in pt-spec.txt section "3.3.5 Pluggable
              Transport Status Message".
```

This event is triggered when tor receives a log message from the PT.

Example:

```
    PT (obfs4): STATUS TRANSPORT=obfs4 CONNECT=Success
```

The resulting control port event would be:

```
    650 PT_STATUS PT=/usr/bin/obs4proxy TRANSPORT=obfs4
CONNECT=Success
```

[PT_STATUS was added in Tor 0.4.0.1-alpha]

Once TLS is finished with a relay, Tor will send a CREATE_FAST cell to establish a one-hop circuit for retrieving directory information. It will remain in this phase until it receives the CREATED_FAST cell back, indicating that the circuit is ready.

Phase 20: tag=requesting_status summary="Asking for networkstatus consensus"

Once we've finished our one-hop circuit, we will start a new stream for fetching the networkstatus consensus. We'll stay in this phase until we get the RELAY_CONNECTED message back, indicating that we've established a directory connection.

Phase 25: tag=loading_status summary="Loading networkstatus consensus"

Once we've established a directory connection, we will start fetching the networkstatus consensus document. This could take a while; this phase is a good opportunity for using the "progress" keyword to indicate partial progress.

This phase could stall if the directory server we picked doesn't have a copy of the networkstatus consensus so we have to ask another, or it does give us a copy but we don't find it valid.

Phase 40: tag=loading_keys summary="Loading authority key certs"

Sometimes when we've finished loading the networkstatus consensus, we find that we don't have all the authority key certificates for the keys that signed the consensus. At that point we put the consensus we fetched on hold and fetch the keys so we can verify the signatures.

```
   Phase 45
   tag=requesting_descriptors summary="Asking for relay descriptors
                                       [ for internal paths]"
```

Once we have a valid networkstatus consensus and we've checked all its signatures, we start asking for relay descriptors. We stay in this phase until we have received a RELAY_CONNECTED message in response to a request for descriptors.

```
 [Newer versions of Tor (0.2.6.2-alpha and later):
  If the consensus contains Exits (the typical case), Tor will build
both
  exit and internal circuits. When bootstrap completes, Tor will be
ready
  to handle an application requesting an exit circuit to services like
the
  World Wide Web.
```

If the consensus does not contain Exits, Tor will only build internal circuits. In this case, earlier statuses will have included “internal” as indicated above. When bootstrap completes, Tor will be ready to handle an application requesting an internal circuit to hidden services at “.onion” addresses.

If a future consensus contains Exits, exit circuits may become available.]

Phase 0: tag=starting summary=“Starting”

Tor starts out in this phase.

Phase 5: tag=conn_dir summary=“Connecting to directory server”

Tor sends this event as soon as Tor has chosen a directory server – e.g. one of the authorities if bootstrapping for the first time or after a long downtime, or one of the relays listed in its cached directory information otherwise.

Tor will stay at this phase until it has successfully established a TCP connection with some directory server. Problems in this phase generally happen because Tor doesn’t have a network connection, or because the local firewall is dropping SYN packets.

Phase 10: tag=handshake_dir summary=“Finishing handshake with directory server”

This event occurs when Tor establishes a TCP connection with a relay or authority used as a directory server (or its https proxy if it’s using one). Tor remains in this phase until the TLS handshake with the relay or authority is finished.

Problems in this phase generally happen because Tor’s firewall is doing more sophisticated MITM attacks on it, or doing packet-level keyword recognition of Tor’s handshake.

Phase 15: tag=onehop_create summary=“Establishing an encrypted directory connection”

Implementation notes

Authentication

If the control port is open and no authentication operation is enabled, Tor trusts any local user that connects to the control port. This is generally a poor idea.

If the ‘CookieAuthentication’ option is true, Tor writes a “magic cookie” file named “control_auth_cookie” into its data directory (or to another file specified in the ‘CookieAuthFile’ option). To authenticate, the controller must demonstrate that it can read the contents of the cookie file:

- Current versions of Tor support cookie authentication

```
    using the "COOKIE" authentication method: the controller sends the
    contents of the cookie file, encoded in hexadecimal.  This
    authentication method exposes the user running a controller to an
    unintended information disclosure attack whenever the controller
    has greater filesystem read access than the process that it has
    connected to.  (Note that a controller may connect to a process
    other than Tor.)  It is almost never safe to use, even if the
    controller's user has explicitly specified which filename to read
    an authentication cookie from.  For this reason, the COOKIE
    authentication method has been deprecated and will be removed from
    Tor before some future version of Tor.

  * 0.2.2.x versions of Tor starting with 0.2.2.36, and all versions
of

    Tor after 0.2.3.12-alpha, support cookie authentication using the
    "SAFECOOKIE" authentication method, which discloses much less
    information about the contents of the cookie file.
```

If the ‘HashedControlPassword’ option is set, it must contain the salted hash of a secret password. The salted hash is computed according to the S2K algorithm in RFC 2440 (OpenPGP), and prefixed with the s2k specifier. This is then encoded in hexadecimal, prefixed by the indicator sequence “16:”. Thus, for example, the password ‘foo’ could encode to:

```
      16:660537E3E1CD49996044A3BF558097A981F539FEA2F9DA662B4626C1C2
         ++++++++++++++++^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
            salt                     hashed value
                       indicator
```

You can generate the salt of a password by calling

```
'tor --hash-password <password>'
```

or by using the example code in the Python and Java controller libraries. To authenticate under this scheme, the controller sends Tor the original secret that was used to generate the password, either as a quoted string or encoded in hexadecimal.

Don't let the buffer get too big

With old versions of Tor (before 0.2.0.16-alpha), if you ask for lots of events, and 16MB of them queue up on the buffer, the Tor process will close the socket.

Newer Tor versions do not have this 16 MB buffer limit. However, if you leave huge numbers of events unread, Tor may still run out of memory, so you should still be careful about buffer size.

Backward compatibility with v0 control protocol

The 'version 0' control protocol was replaced in Tor 0.1.1.x. Support was removed in Tor 0.2.0.x. Every non-obsolete version of Tor now supports the version 1 control protocol.

For backward compatibility with the "version 0" control protocol, Tor used to check whether the third octet of the first command is zero. (If it was, Tor assumed that version 0 is in use.)

This compatibility was removed in Tor 0.1.2.16 and 0.2.0.4-alpha.

Tor config options for use by controllers

Tor provides a few special configuration options for use by controllers. These options are not saved to disk by SAVECONF. Most can be set and examined by the SETCONF and GETCONF commands, but some (noted below) can only be given in a torrc file or on the command line.

Generally, these options make Tor unusable by disabling a portion of Tor's normal operations. Unless a controller provides replacement functionality to fill this gap, Tor will not correctly handle user requests.

This is similar to conn_done, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 89:
   tag=ap_handshake summary="Finishing handshake with a relay to build
 circuits"
  [New in 0.4.0.x]
```

This is similar to handshake, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 90:
   tag=ap_handshake_done summary="Handshake finished with a relay to
 build circuits"
  [New in 0.4.0.x]
```

This is similar to handshake_done, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 95:
   tag=circuit_create summary="Establishing a[n internal] Tor circuit"
  [prior to 0.4.0.x, this was numbered 90]
```

Once we've finished our TLS handshake with the first hop of a circuit, we will set about trying to make some 3-hop circuits in case we need them soon.

```
  [Some versions of Tor (starting with 0.2.6.2-alpha but before
   0.4.0.x): Tor could report having internal paths only; see Section
   5.6]
```

Phase 100: tag=done summary="Done"

A full 3-hop circuit has been established. Tor is ready to handle application connections now.


```
  [Some versions of Tor (starting with 0.2.6.2-alpha but before
   0.4.0.x): Tor could report having internal paths only; see Section
   5.6]
```


Bootstrap phases reported by older versions of Tor

These phases were reported by Tor older than 0.4.0.x. For newer versions of Tor, see Section 5.5.

```
   Phase 76:
   tag=ap_conn_pt summary="Connecting to pluggable transport to build
   circuits"
  [New in 0.4.0.x]
```

This is similar to conn_pt, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 77:
   tag=ap_conn_done_pt summary="Connected to pluggable transport to
 build circuits"
  [New in 0.4.0.x]
```

This is similar to conn_done_pt, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 78:
   tag=ap_conn_proxy summary="Connecting to proxy to build circuits"
  [New in 0.4.0.x]
```

This is similar to conn_proxy, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 79:
   tag=ap_conn_done_proxy summary="Connected to proxy to build
 circuits"
  [New in 0.4.0.x]
```

This is similar to conn_done_proxy, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 80:
   tag=ap_conn summary="Connecting to a relay to build circuits"
  [New in 0.4.0.x]
```

This is similar to conn, except for making connections to additional relays or bridges that Tor needs to use to build application circuits.

```
   Phase 85:
   tag=ap_conn_done summary="Connected to a relay to build circuits"
  [New in 0.4.0.x]
```

```
    If true, Tor will try to launch all directory operations through
    anonymous connections.  (Ordinarily, Tor only tries to anonymize
    requests related to hidden services.)  This option will slow down
    directory access, and may stop Tor from working entirely if it
does not
    yet have enough directory information to build circuits.

    (Boolean. Default: "0".)

  __DisablePredictedCircuits

    If true, Tor will not launch preemptive "general-purpose" circuits
for
    streams to attach to.  (It will still launch circuits for testing
and
    for hidden services.)

    (Boolean. Default: "0".)

  __LeaveStreamsUnattached

    If true, Tor will not automatically attach new streams to
circuits;
    instead, the controller must attach them with ATTACHSTREAM.  If
the
    controller does not attach the streams, their data will never be
routed.

    (Boolean. Default: "0".)

  __HashedControlSessionPassword

    As HashedControlPassword, but is not saved to the torrc file by
    SAVECONF.  Added in Tor 0.2.0.20-rc.

  __ReloadTorrcOnSIGHUP

    If this option is true (the default), we reload the torrc from
disk
    every time we get a SIGHUP (from the controller or via a signal).
    Otherwise, we don't.  This option exists so that controllers can
keep
    their options from getting overwritten when a user sends Tor a HUP
for
    some other reason (for example, to rotate the logs).

    (Boolean.  Default: "1")

  __OwningControllerProcess

    If this option is set to a process ID, Tor will periodically check
    whether a process with the specified PID exists, and exit if one
    does not.  Added in Tor 0.2.2.28-beta.  This option's intended use
    is documented in section 3.23 with the related TAKEOWNERSHIP
```

This phase could stall if the directory server we picked doesn't have a copy of the networkstatus consensus so we have to ask another, or it does give us a copy but we don't find it valid.

Phase 40: tag=loading_keys summary="Loading authority key certs"

Sometimes when we've finished loading the networkstatus consensus, we find that we don't have all the authority key certificates for the keys that signed the consensus. At that point we put the consensus we fetched on hold and fetch the keys so we can verify the signatures.

Phase 45 tag=requesting_descriptors summary="Asking for relay descriptors"

Once we have a valid networkstatus consensus and we've checked all its signatures, we start asking for relay descriptors. We stay in this phase until we have received a RELAY_CONNECTED message in response to a request for descriptors.

```
[Some versions of Tor (starting with 0.2.6.2-alpha but before
 0.4.0.x): Tor could report having internal paths only; see Section
 5.6]
```

Phase 50: tag=loading_descriptors summary="Loading relay descriptors"

We will ask for relay descriptors from several different locations, so this step will probably make up the bulk of the bootstrapping, especially for users with slow connections. We stay in this phase until we have descriptors for a significant fraction of the usable relays listed in the networkstatus consensus (this can be between 25% and 95% depending on Tor's configuration and network consensus parameters). This phase is also a good opportunity to use the "progress" keyword to indicate partial steps.

```
[Some versions of Tor (starting with 0.2.6.2-alpha but before
 0.4.0.x): Tor could report having internal paths only; see Section
 5.6]

 Phase 75:
 tag=enough_dirinfo summary="Loaded enough directory info to build
 circuits"
[New in 0.4.0.x; previously, Tor would misleadingly report the
 "conn_or" tag once it had enough directory info.]
```

```
   Phase 10:
   tag=conn_done summary="Connected to a relay"
  [New in 0.4.0.x]

   Tor has completed its first connection to a relay.

   Phase 14:
   tag=handshake summary="Handshaking with a relay"
  [New in 0.4.0.x; prior versions of Tor had a "handshake_dir" phase]

   Tor is in the process of doing a TLS handshake with a relay.

   Phase 15:
   tag=handshake_done summary="Handshake with a relay done"
  [New in 0.4.0.x]

   Tor has completed its TLS handshake with a relay.
```

Phases in Bootstrap Stage 2

```
   Phase 20:
   tag=onehop_create summary="Establishing an encrypted directory
 connection"
  [prior to 0.4.0.x, this was numbered 15]
```

Once TLS is finished with a relay, Tor will send a CREATE_FAST cell to establish a one-hop circuit for retrieving directory information. It will remain in this phase until it receives the CREATED_FAST cell back, indicating that the circuit is ready.

```
   Phase 25:
   tag=requesting_status summary="Asking for networkstatus consensus"
  [prior to 0.4.0.x, this was numbered 20]
```

Once we've finished our one-hop circuit, we will start a new stream for fetching the networkstatus consensus. We'll stay in this phase until we get the RELAY_CONNECTED message back, indicating that we've established a directory connection.

```
   Phase 30:
   tag=loading_status summary="Loading networkstatus consensus"
  [prior to 0.4.0.x, this was numbered 25]
```

Once we've established a directory connection, we will start fetching the networkstatus consensus document. This could take a while; this phase is a good opportunity for using the "progress" keyword to indicate partial progress.

```
    command.

    Note that this option can only specify a single process ID, unlike
    the TAKEOWNERSHIP command which can be sent along multiple control
    connections.

    (String.  Default: unset.)

  __OwningControllerFD

    If this option is a valid socket, Tor will start with an open
control
    connection on this socket.  Added in Tor 0.3.3.1-alpha.

    This socket will be an owning controller, as if it had already
called
    TAKEOWNERSHIP.  It will be automatically authenticated.  This
option
    should only be used by other programs that are starting Tor.

    This option cannot be changed via SETCONF; it must be set in a
torrc or
    via the command line.

    (Integer. Default: -1.)

  __DisableSignalHandlers

    If this option is set to true during startup, then Tor will not
install
    any signal handlers to watch for POSIX signals.  The SIGNAL
controller
    command will still work.

    This option is meant for embedding Tor inside another process,
when
    the controlling process would rather handle signals on its own.

    This option cannot be changed via SETCONF; it must be set in a
torrc or
    via the command line.

    (Boolean. Default: 0.)
```

Phases from the Bootstrap status event

```
  [For the bootstrap phases reported by Tor prior to 0.4.0.x, see
   Section 5.6.]
```

This section describes the various bootstrap phases currently reported by Tor. Controllers should not assume that the percentages and tags listed here will continue to match up, or even that the tags will stay in the same order. Some phases might also be skipped (not reported) if the associated bootstrap step is already complete, or if the phase no longer is necessary. Only “starting” and “done” are guaranteed to exist in all future versions.

Current Tor versions enter these phases in order, monotonically. Future Tors MAY revisit earlier phases, for example, if the network fails.

Overview of Bootstrap reporting

Bootstrap phases can be viewed as belonging to one of three stages:

1. Initial connection to a Tor relay or bridge
2. Obtaining directory information
3. Building an application circuit

Tor doesn’t specifically enter Stage 1; that is a side effect of other actions that Tor is taking. Tor could be making a connection to a fallback directory server, or it could be making a connection to a guard candidate. Either one counts as Stage 1 for the purposes of bootstrap reporting.

Stage 2 might involve Tor contacting directory servers, or it might involve reading cached directory information from a previous session. Large parts of Stage 2 might be skipped if there is already enough cached directory information to build circuits. Tor will defer reporting progress in Stage 2 until Stage 1 is complete.

Tor defers this reporting because Tor can already have enough directory information to build circuits, yet not be able to connect to a relay. Without that deferral, a user might misleadingly see Tor stuck at a large amount of progress when something as fundamental as making a TCP connection to any relay is failing.

Tor also doesn’t specifically enter Stage 3; that is a side effect of Tor building circuits for some purpose or other. In a typical client, Tor builds predicted circuits to provide lower latency for application connection requests. In Stage 3, Tor might make new connections to relays or bridges that it did not connect to in Stage 1.

Phases in Bootstrap Stage 1

Phase 0: tag=starting summary=“Starting”

Tor starts out in this phase.

```
   Phase 1:
   tag=conn_pt summary="Connecting to pluggable transport"
  [This phase is new in 0.4.0.x]
```

Tor is making a TCP connection to the transport plugin for a pluggable transport. Tor will use this pluggable transport to make its first connection to a bridge.

```
   Phase 2:
   tag=conn_done_pt summary="Connected to pluggable transport"
  [New in 0.4.0.x]
```

Tor has completed its TCP connection to the transport plugin for the pluggable transport.

```
   Phase 3:
   tag=conn_proxy summary="Connecting to proxy"
  [New in 0.4.0.x]
```

Tor is making a TCP connection to a proxy to make its first connection to a relay or bridge.

```
   Phase 4:
   tag=conn_done_proxy summary="Connected to proxy"
  [New in 0.4.0.x]
```

Tor has completed its TCP connection to a proxy to make its first connection to a relay or bridge.

```
   Phase 5:
   tag=conn summary="Connecting to a relay"
  [New in 0.4.0.x; prior versions of Tor had a "conn_dir" phase that
   sometimes but not always corresponded to connecting to a directory
 server]
```

Tor is making its first connection to a relay. This might be through a pluggable transport or proxy connection that Tor has already established.