Tor Metrics (/)

Welcome to CollecTor, your friendly data-collecting service in the Tor network

CollecTor fetches data from various nodes and services in the public Tor network and makes it available to the world. If you're doing research on the Tor network, or if you're developing an application that uses Tor network data, this is your place to start.

> Browse Recent Descriptors (/collector/recent/)

> Browse Archived Descriptors (/collector/archive/)

Available Descriptors

Descriptors are available in two different file formats: recent descriptors that were published in the last 72 hours are available as plain text, and archived descriptors covering over 10 years of Tor network history are available as compressed tarballs.

Descriptor Type	Type Annotation	Descriptors
Tor Relay Descriptors		
Relay Server Descriptors	`@type server-descriptor 1.0`	> recent (/collector/recent/relay-descriptors/server-descriptors/) > archive (/collector/archive/relay-descriptors/server-descriptors/)
Relay Extra-info Descriptors	`@type extra-info 1.0`	> recent (/collector/recent/relay-descriptors/extra-infos/) > archive (/collector/archive/relay-descriptors/extra-infos/)
Network Status Consensuses	`@type network-status-consensus-3 1.0`	> recent (/collector/recent/relay-descriptors/consensuses/) > archive (/collector/archive/relay-descriptors/consensuses/)
Network Status Votes	`@type network-status-vote-3 1.0`	> recent (/collector/recent/relay-descriptors/votes/) > archive (/collector/archive/relay-descriptors/votes/)
Directory Key Certificates	`@type dir-key-certificate-3 1.0`	> archive (/collector/archive/relay-descriptors/)

Descriptor Type	Type Annotation	Descriptors
Microdescriptor Consensuses	`@type network-status-microdesc-consensus-3 1.0`	recent (/collector/recent/relay-descriptors/microdescs/consensus-microdesc/) archive (/collector/archive/relay-descriptors/microdescs/)
Microdescriptors	`@type microdescriptor 1.0`	recent (/collector/recent/relay-descriptors/microdescs/micro/) archive (/collector/archive/relay-descriptors/microdescs/)
Version 2 Network Statuses	`@type network-status-2 1.0`	archive (/collector/archive/relay-descriptors/statuses/)
Version 1 Directories	`@type directory 1.0`	archive (/collector/archive/relay-descriptors/tor/)
Tor Bridge Descriptors		
Bridge Network Statuses	`@type bridge-network-status 1.2`	recent (/collector/recent/bridge-descriptors/statuses/) archive (/collector/archive/bridge-descriptors/statuses/)
Bridge Server Descriptors	`@type bridge-server-descriptor 1.2`	recent (/collector/recent/bridge-descriptors/server-descriptors/) archive (/collector/archive/bridge-descriptors/server-descriptors/)
Bridge Extra-info Descriptors	`@type bridge-extra-info 1.3`	recent (/collector/recent/bridge-descriptors/extra-infos/) archive (/collector/archive/bridge-descriptors/extra-infos/)
Tor Hidden Service Descriptors		
Hidden Service Descriptors	`@type hidden-service-descriptor 1.0`	
BridgeDB's Bridge Pool Assignments		
Bridge Pool Assignments	`@type bridge-pool-assignment 1.0`	archive (/collector/archive/bridge-pool-assignments/)
TorDNSEL's Exit Lists		
Exit Lists	`@type tordnsel 1.0`	recent (/collector/recent/exit-lists/) archive (/collector/archive/exit-lists/)
Torperf's and OnionPerf's Performance Data		
Torperf Measurement Results	`@type torperf 1.1`	recent (/collector/recent/torperf/) archive (/collector/archive/torperf/)
Tor web server logs		
Tor web server logs		recent (/collector/recent/webstats/) archive (/collector/archive/webstats/)

Data Formats

Each descriptor provided here contains an `@type` annotation using the format `@type $descriptortype $major.$minor`. Any tool that processes these descriptors may parse files without meta data or with an unknown descriptor type at its own risk, can safely parse files with known descriptor type and same major version number, and should not parse files with known descriptor type and higher major version number.

Tor Relay Descriptors

Relays and directory authorities publish relay descriptors, so that clients can select relays for their paths through the Tor network. All these relay descriptors are specified in the Tor directory protocol, version 3 (https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt) specification document (or in the earlier protocol version 2 (https://gitweb.torproject.org/torspec.git/tree/attic/dir-spec-v2.txt) or version 1 (https://gitweb.torproject.org/torspec.git/tree/attic/dir-spec-v1.txt)).

Relay Server Descriptors `@type server-descriptor 1.0`

> recent (/collector/recent/relay-descriptors/server-descriptors/)

> archive (/collector/archive/relay-descriptors/server-descriptors/)

Server descriptors contain information that relays publish about themselves. Tor clients once downloaded this information, but now they use microdescriptors instead.

Recently collected files follow the naming schema `YYYY-MM-DD-HH-MM-SS-server-descriptors` with `YYYY-MM-DD-HH-MM-SS` being the download start time and contain all collected descriptors concatenated into a single file.

Archive tarballs follow the naming schema `server-descriptors-YYYY-MM.tar.xz` with `YYYY-MM` being year and month of the descriptor publication time. Archived files follow the naming schema `server-descriptors-YYYY-MM/X/Y/DIGEST` with `YYYY-MM` again being year and month of the descriptor publication time, `X` and `Y` being the first and second character of the hex-encoded, lower-case SHA-1 descriptor digest, and `DIGEST` being that descriptor digest in full.

Relay Extra-info Descriptors `@type extra-info 1.0`

> recent (/collector/recent/relay-descriptors/extra-infos/)

> archive (/collector/archive/relay-descriptors/extra-infos/)

Extra-info descriptors contain relay information that Tor clients do not need in order to function. These are self-published, like server descriptors, but not downloaded by clients by default. The extra-info descriptors in the descriptor archives contain one descriptor per file, whereas the recently published files contain all descriptors collected in an hour concatenated into a single file.

Network Status Consensuses `@type network-status-consensus-3 1.0`

> recent (/collector/recent/relay-descriptors/consensuses/)

> archive (/collector/archive/relay-descriptors/consensuses/)

Though Tor relays are decentralized, the directories that track the overall network are not. These central points are called directory authorities, and every hour they publish a document called a consensus, or network status document. The consensus is made up of router status entries containing flags, heuristics used for relay selection, etc.

Recently collected files follow the naming schema `YYYY-MM-DD-HH-MM-SS-consensus` with `YYYY-MM-DD-HH-MM-SS` being the network status valid-after time.

Archive tarballs follow the naming schema `consensuses-YYYY-MM.tar.xz` with YYYY-MM being year and month of the network status valid-after time. Archived files follow the naming schema `consensuses-YYYY-MM/DD/YYYY-MM-DD-HH-MM-SS-consensus` with `YYYY-MM/DD/YYYY-MM-DD-HH-MM-SS` again being the network status valid-after time.

Network Status Votes `@type network-status-vote-3 1.0`

> recent (/collector/recent/relay-descriptors/votes/)

> archive (/collector/archive/relay-descriptors/votes/)

The directory authorities exchange votes every hour to come up with a common consensus. Vote documents are by far the largest documents provided here.

Directory Key Certificates `@type dir-key-certificate-3 1.0` > archive (/collector/archive/relay-descriptors/)

The directory authorities sign votes and the consensus with their key that they publish in a key certificate. These key certificates change once every few months, so they are only available in a single descriptor archive tarball.

Microdescriptor Consensuses `@type network-status-microdesc-consensus-3 1.0`

> recent (/collector/recent/relay-descriptors/microdescs/) > archive (/collector/archive/relay-descriptors/microdescs/)

Tor clients used to download all server descriptors of active relays, but now they only download the smaller microdescriptors which are derived from server descriptors. The microdescriptor consensus lists all active relays and references their currently used microdescriptor. The descriptor archive tarballs contain both microdescriptor consensuses and referenced microdescriptors together.

Microdescriptors `@type microdescriptor 1.0` > recent (/collector/recent/relay-descriptors/microdescs/)

> archive (/collector/archive/relay-descriptors/microdescs/)

Microdescriptors are minimalistic documents that just includes the information necessary for Tor clients to work. The descriptor archive tarballs contain both microdescriptor consensuses and referenced microdescriptors together. The microdescriptors in descriptor archive tarballs contain one descriptor per file, whereas the recently published files contain all descriptors collected in an hour concatenated into a single file.

Version 2 Network Statuses `@type network-status-2 1.0` > archive (/collector/archive/relay-descriptors/statuses/)

Version 2 network statuses have been published by the directory authorities before consensuses have been introduced. In contrast to consensuses, each directory authority published their own authoritative view on the network, and clients combined these documents locally. We stopped archiving version 2 network statuses in 2012.

Version 1 Directories `@type directory 1.0` > archive (/collector/archive/relay-descriptors/tor/)

The first directory protocol version combined the list of active relays with server descriptors in a single directory document. We stopped archiving version 1 directories in 2007.

Tor Bridge Descriptors

Bridges and the bridge authority publish bridge descriptors that are used by censored clients to connect to the Tor network. We cannot, however, make bridge descriptors available as we do with relay descriptors, because that would defeat the purpose of making bridges hard to enumerate for censors. We therefore sanitize bridge descriptors by removing all potentially identifying information and publish sanitized versions here. The sanitizing steps are specified in detail on a separate page (bridge-descriptors.html).

Bridge Network Statuses `@type bridge-network-status 1.2` > recent (/collector/recent/bridge-descriptors/statuses/)

> archive (/collector/archive/bridge-descriptors/statuses/)

Sanitized bridge network statuses are similar to version 2 relay network statuses, but with only a `published` line and a `fingerprint` line in the header, and without any lines in the footer. The format has changed over time to accomodate changes to the sanitizing process, with earlier versions being:

- `@type bridge-network-status 1.0` was the first version.
- `@type bridge-network-status 1.1` introduced sanitized TCP ports.
- `@type bridge-network-status 1.2` introduced the `fingerprint` line, containing the fingerprint of the bridge authority which produced the document, to the header.

Bridge Server descriptors `@type bridge-server-descriptor 1.2`

> recent (/collector/recent/bridge-descriptors/server-descriptors/) > archive (/collector/archive/bridge-descriptors/server-descriptors/)

Bridge server descriptors follow the same format as relay server descriptors, except for the sanitizing steps described above. The bridge server descriptor archive tarballs contain one descriptor per file, whereas recently published bridge server descriptor files contain all descriptors collected in an hour concatenated into a single file to reduce the number of files. The format has changed over time to accomodate changes to the sanitizing process, with earlier versions being:

- `@type bridge-server-descriptor 1.0` was the first version.
- There was supposed to be a newer version indicating added `ntor-onion-key` lines, but due to a mistake only the version number of sanitized bridge extra-info descriptors was raised. As a result, there may be sanitized bridge server descriptors with version `@type bridge-server-descriptor 1.0` with and without those lines.
- `@type bridge-server-descriptor 1.1` added `master-key-ed25519` lines and `router-digest-sha256` to server descriptors published by bridges using an Ed25519 master key.
- `@type bridge-server-descriptor 1.2` introduced sanitized TCP ports.

Bridge Extra-info Descriptors `@type bridge-extra-info 1.3`

> recent (/collector/recent/bridge-descriptors/extra-infos/) > archive (/collector/archive/bridge-descriptors/extra-infos/)

Bridge extra-info descriptors follow the same format as relay extra-info descriptors, except for the sanitizing steps described above. The format has changed over time to accomodate changes to the sanitizing process, with earlier versions being:

- `@type bridge-extra-info 1.0` was the first version.
- `@type bridge-extra-info 1.1` added sanitized `transport` lines.
- `@type bridge-extra-info 1.2` was supposed to indicate added `ntor-onion-key` lines, but those changes only affect bridge server descriptors, not extra-info descriptors. So, nothing has changed as compared to version 1.1.
- `@type bridge-extra-info 1.3` added `master-key-ed25519` lines and `router-digest-sha256` to extra-info descriptors published by bridges using an Ed25519 master key.

The bridge extra-info descriptor archive tarballs contain one descriptor per file, whereas recently published bridge extra-info descriptor files contain all descriptors collected in an hour concatenated into a single file to reduce the number of files.

Tor Hidden Service Descriptors

Tor hidden services make it possible for users to hide their locations while offering various kinds of services, such as web publishing or an instant messaging server. A hidden service assembles a hidden service descriptor to make its service available in the network. This descriptor gets stored on hidden service directories and can be retrieved by hidden service clients. Hidden service descriptors are not formally archived, but some libraries support parsing these descriptors when obtaining them from a locally running Tor instance.

Hidden Service Descriptors `@type hidden-service-descriptor 1.0`

Hidden service descriptors contain all details that are necessary for clients to connect to a hidden service. Despite the version number being 1.0, these descriptors are part of the version 2 hidden service protocol.

BridgeDB's Bridge Pool Assignments

The bridge distribution service BridgeDB publishes bridge pool assignments describing which bridges it has assigned to which distribution pool. BridgeDB receives bridge network statuses from the bridge authority, assigns these bridges to persistent distribution rings, and hands them out to bridge users. BridgeDB periodically dumps the list of running bridges with information about the rings, subrings, and file buckets to which they are assigned to a local file. The sanitized versions of these lists containing SHA-1 hashes of bridge fingerprints instead of the original fingerprints are available for statistical analysis.

Bridge Pool Assignments `@type bridge-pool-assignment 1.0`

> archive (/collector/archive/bridge-pool-assignments/)

The document below shows a BridgeDB pool assignment file from March 13, 2011. Every such file begins with a line containing the timestamp when BridgeDB wrote this file. Subsequent lines start with the SHA-1 hash of a bridge fingerprint, followed by ring, subring, and/or file bucket information. There are currently three distributor ring types in BridgeDB:

1. **unallocated:** These bridges are not distributed by BridgeDB, but are either reserved for manual distribution or are written to file buckets for distribution via an external tool. If a bridge in the `unallocated` ring is assigned to a file bucket, this is noted by `bucket=$bucketname`.
2. **email:** These bridges are distributed via an e-mail autoresponder. Bridges can be assigned to subrings by their OR port or relay flag which is defined by `port=$port` and/or `flag=$flag`.
3. **https:** These bridges are distributed via https server. There are multiple https rings to further distribute bridges by IP address ranges, which is denoted by `ring=$ring`. Bridges in the `https` ring can also be assigned to subrings by OR port or relay flag which is defined by `port=$port` and/or `flag=$flag`.

```
bridge-pool-assignment 2011-03-13 14:38:03
00b834117566035736fc6bd4ece950eace8e057a unallocated
00e923e7a8d87d28954fee7503e480f3a03ce4ee email port=443 flag=stable
0103bb5b00ad3102b2dbafe9ce709a0a7c1060e4 https ring=2 port=443 flag=stable
[...]
```

As of December 8, 2014, bridge pool assignment files are no longer archived.

TorDNSEL's Exit Lists

The exit list service TorDNSEL (https://gitweb.torproject.org/tordnsel.git/tree/README) publishes exit lists containing the IP addresses of relays that it found when exiting through them.

Exit Lists `@type tordnsel 1.0` > recent (/collector/recent/exit-lists/) > archive (/collector/archive/exit-lists/)

TorDNSEL makes the list of known exits and corresponding exit IP addresses available in a specific format. The document below shows an entry of the exit list written on December 28, 2010 at 15:21:44 UTC. This entry means that the relay with fingerprint `63BA..` which published a descriptor at 07:35:55 and was contained in a version 2 network status from 08:10:11 uses two different IP addresses for exiting. The first address `91.102.152.236` was found in a test performed at 07:10:30. When looking at the corresponding server descriptor, one finds that this is also the IP address on which the relay accepts connections from inside the Tor network. A second test performed at 10:35:30 reveals that the relay also uses IP address `91.102.152.227` for exiting.

```
ExitNode 63BA28370F543D175173E414D5450590D73E22DC
Published 2010-12-28 07:35:55
LastStatus 2010-12-28 08:10:11
ExitAddress 91.102.152.236 2010-12-28 07:10:30
ExitAddress 91.102.152.227 2010-12-28 10:35:30
```

Torperf's and OnionPerf's Performance Data

The performance measurement services Torperf and OnionPerf publish performance data from making simple HTTP requests over the Tor network. Torperf/OnionPerf use a SOCKS client to download files of various sizes over the Tor network and notes how long substeps take.

Torperf and OnionPerf Measurement Results `@type torperf 1.1` > recent (/collector/recent/torperf/) > archive (/collector/archive/torperf/)

A Torperf or OnionPerf results file contains a single line per Torperf/OnionPerf run with `key=value` pairs. Such a result line is sufficient to learn about 1) the Tor and Torperf/OnionPerf configuration, 2) measurement results, and 3) additional

information that might help explain the results. Known keys in `@type torperf 1.0` are explained below.

- Configuration
  - `SOURCE:` Configured name of the data source; required.
  - `FILESIZE:` Configured file size in bytes; required.
  - Other meta data describing the Tor or Torperf/OnionPerf configuration, e.g., GUARD for a custom guard choice; optional.
- Measurement results
  - `START:` Time when the connection process starts; required.
  - `SOCKET:` Time when the socket was created; required.
  - `CONNECT:` Time when the socket was connected; required.
  - `NEGOTIATE:` Time when SOCKS 5 authentication methods have been negotiated; required.
  - `REQUEST:` Time when the SOCKS request was sent; required.
  - `RESPONSE:` Time when the SOCKS response was received; required.
  - `DATAREQUEST:` Time when the HTTP request was written; required.
  - `DATARESPONSE:` Time when the first response was received; required.
  - `DATACOMPLETE:` Time when the payload was complete; required.
  - `WRITEBYTES:` Total number of bytes written; required.
  - `READBYTES:` Total number of bytes read; required.
  - `DIDTIMEOUT:` 1 if the request timed out, 0 otherwise; optional.
  - `DATAPERCx:` Time when x% of expected bytes were read for x = { 10, 20, 30, 40, 50, 60, 70, 80, 90 }; optional.
  - Other measurement results, e.g., START_RENDCIRC, GOT_INTROCIRC, etc. for hidden-service measurements; optional.
- Additional information
  - `LAUNCH:` Time when the circuit was launched; optional.
  - `USED_AT:` Time when this circuit was used; optional.
  - `PATH:` List of relays in the circuit, separated by commas; optional.
  - `BUILDTIMES:` List of times when circuit hops were built, separated by commas; optional.
  - `TIMEOUT:` Circuit build timeout in milliseconds that the Tor client used when building this circuit; optional.
  - `QUANTILE:` Circuit build time quantile that the Tor client uses to determine its circuit-build timeout; optional.
  - `CIRC_ID:` Circuit identifier of the circuit used for this measurement; optional.
  - `USED_BY:` Stream identifier of the stream used for this measurement; optional.
  - Other fields containing additional information; optional.

OnionPerf adds a few more keys in `@type torperf 1.1` :

- `ENDPOINTLOCAL:` Hostname, IP address, and port that the TGen client used to connect to the local tor SOCKS port, formatted as `hostname:ip:port` , which may be `"NULL:0.0.0.0:0"` if TGen was not able to find this information; optional.
- `ENDPOINTPROXY:` Hostname, IP address, and port that the TGen client used to connect to the SOCKS proxy server that tor runs, formatted as `hostname:ip:port` , which may be `"NULL:0.0.0.0:0"` if TGen was not able to find this information; optional.
- `ENDPOINTREMOTE:` Hostname, IP address, and port that the TGen client used to connect to the remote server, formatted as `hostname:ip:port` , which may be `"NULL:0.0.0.0:0"` if TGen was not able to find this information; optional.
- `HOSTNAMELOCAL:` Client machine hostname, which may be `"(NULL)"` if the TGen client was not able to find this information; optional.
- `HOSTNAMEREMOTE:` Server machine hostname, which may be `"(NULL)"` if the TGen server was not able to find this information; optional.
- `SOURCEADDRESS:` Public IP address of the OnionPerf host obtained by connecting to well-known servers and finding the IP address in the result, which may be `"unknown"` if OnionPerf was not able to find this information; optional.

Tor web server logs

Tor's web servers, like most web servers, keep request logs for maintenance and informational purposes. However, unlike most other web servers, Tor's web servers use a privacy-aware log format that avoids logging too sensitive data about their users. Also unlike most other web server logs, Tor's logs are neither archived nor analyzed before performing a number of post-processing steps to further reduce any privacy-sensitive parts.

Tor web server logs

> recent (/collector/recent/webstats/) > archive (/collector/archive/webstats/)

The data format and sanitizing steps for Tor web server logs are specified in detail on a separate page (web-server-logs.html).

Automated Downloads

There are multiple ways to download descriptors from this site. Of course, the obvious way is to browse the directories and download contained files using your browser. However, this method cannot be automated very well.

Recursive downloads using `wget`

A more elaborate way to automatically download descriptors is to use Unix tools like `wget` which support recursively downloading files from this site. Example:

```
wget --recursive \                    # turn on recursive retrieving
     --reject "index.html*" \         # don't retrieve directory listings
     --no-parent \                    # don't ascend to parent directory
     --no-host-directories \          # don't generate host-prefixed directories
     --directory-prefix descriptors \  # set directory prefix
     https://collector.torproject.org/recent/relay-descriptors/microdescs/
```

Custom downloaders using provided `index.json`

Another automated way to download descriptors is to develop a tool that uses the provided `index.json` (https://collector.torproject.org/index/index.json) file or one of its compressed versions `index.json.gz` (https://collector.torproject.org/index/index.json.gz), `index.json.bz2` (https://collector.torproject.org/index/index.json.bz2), or `index.json.xz` (https://collector.torproject.org/index/index.json.xz). These files contain a machine-readable representation of all descriptor files available on this site. Index files use the following custom JSON data format that might still be extended at a later time:

- Index object: At the document root there is always an index object with the following fields:
  - `"index_created"` : Timestamp when this index was created using pattern `"YYYY-MM-DD HH:MM"` in the UTC timezone.
  - `"build_revision"` : Git revision of the CollecTor instance's software used to create this file, which will be omitted if unknown.
  - `"path"` : Base URL of this index file and all included resources.
  - `"files"` : List of file objects of files available from the document root, which will be omitted if empty.
  - `"directories"` : List of directory objects of directories available from the document root, which will be omitted if empty.
- Directory object: There is one directory object for each directory or subdirectory in the document tree containing similar fields as the index object:
  - `"path"` : Relative path of the directory.
  - `"files"` : List of file objects of files available from this directory, which will be omitted if empty.

    - `"directories"` : List of directory objects of directories available from this directory, which will be omitted if empty.
- File object: Each file that is available in the document tree is represented by a file object with the following fields:
    - `"path"` : Relative path of the file.
    - `"size"` : Size of the file in bytes.
    - `"last_modified"` : Timestamp when the file was last modified using pattern `"YYYY-MM-DD HH:MM"` in the UTC timezone.
