NAME

http-analyze – a fast log analyzer for web servers

SYNOPSIS

http-analyze [**–{hdmV}**] [**–3aefgnqvxy**] [**–c** *cfgfile*] [**–l** *libdir*] [**–o** *outdir*]
[**–p** *privdir*] [**–s** *subopt,...*] [**–t** *num,...*] [**–u** *time*] [**–w** *hits*]
[**–F** *format*] [**–G** *suffix,...*] [**–H** *idxfile,...*] [**–I** *date*] [**–E** *date*]
[**–O** *virtname,...*] [**–P** *prolog*] [**–R** *docroot*] [**–S** *srvname*]
[**–T** *TLDfile*] [**–U** *srvurl*] [**–W** *3Dwin*] [*logfile* [...]]

DESCRIPTION

http-analyze analyzes the logfile of a web server and creates a detailed summary of the servers's access load in graphical, tabular, and three-dimensional form. In auto-sense mode (default), **http-analyze** recognizes the logfile format automatically. Supportet formats for logfiles are the *Common Logfile Format (CLF)* and two forms of the so-called *Extended Logfile Format (ELF)*, which is basically the CLF plus user-agent and referrer URL information. All web servers support at least the *Common Logfile Format* and most of them can be configured to produce the *Extended Logfile Format*.

http-analyze has been highly optimized to process large logfiles at maximum speed. There are two modes of operation with different levels of detail in the logfile analysis:

Short statistics ("daily" mode, option **–d**):

> **http-analyze** generates a short summary of the server usage per day for the current month. In this mode, it uses a history file to skip entries which have been processed already. By avoiding detailed analysis of the logfile entries, **http-analyze** requires only a fraction of the time which would be required to generate a full statistics report.

Full statistics ("monthly" mode, option **–m**):

> In full statistics mode, the analyzer generates a complete report for a whole month, which contains much more details than the short statistics report. The history file is used only to produce a summary for the last 12 months without having to analyze the logfiles for those previous periods again. In full statistics mode the actual period to analyze is determined by analyzing the timestamps of the first and last logfile entry read. This is the default if no mode is specified explicitely.

Usually you run **http-analyze** in full statistics mode only, since this report also includes all the information available in short statistics. However, if your logfiles are rather large and if the analyzer causes significant load while generating the full statistics report, you could run it more frequently in short statistics mode with update intervals in the range of 30 minutes to some hours to create an up-to-date report, and then run it in full statistics mode less often, for example once per day or week, to generate a detailed report. The operation modes have been named after their periods covered, namely *daily* for the short and *monthly* for the full statistics mode.

Note that in full statistics mode the analyzer needs to process all logfile entries since the beginning of the current month, while in short statistics mode it skips all entries up to the current day if it finds a valid history. Therefore you should rotate the logfile at the first day of a new month and then generate a final statistics report for the previous month using the logfile just rotated.

If disk space is a concern, you can set up a scheme where the logfiles are rotated and compressed using some compression program once per week or even once per day. In this case, you have to concatenate all logfiles for this month in order of ascending date before feeding them into the analyzer to have it generate a full statistics report. On the first day of the new month, if a detailed report for the previous month has been generated, you can save the corresponding logfile(s) somewhere and finally remove it or them from your production system.

LOGFILE FORMATS

http-analyze recognizes three logfile formats, which can be configured in most web servers:

Common Logfile Format (CLF)

The *Common Logfile Format* is supported by all web servers. The entries contain the following information:

```
dns-name - auth-user [date] "clf-request" clf-status ct-length
```

where the fields have following meaning:

dns-name The IP number of the system accessing the web server. If there is an entry in the *Domain Name System (DNS)* for this IP number and the web server is configured to do DNS lookups, the corresponding hostname is logged instead.

– Unused.

auth-user The username provided by the client to access files which require authentication.

[date] The date of the access as `[DD/MMM/YYYY:HH:MM:SS ±ZZZZ]`.

clf-request The request in format `"method URI proto"`, where *method* is one of **GET**, **HEAD**, **POST**, **PUT**, **BROWSE**, **OPTIONS**, **DELETE** or **TRACE**; *URI* is the *Uniform Resource Identifier*, and *proto* is the protocol parameter containing the HTTP version. The *clf-request* field is surrounded by double quotes.

clf-status This is the (numerical) response code from the server.

ct-length Depending on the server, this number is either the size of the document or the data actually sent over the wire.

Following is an example for an entry in *Common Logfile Format*:

```
car.4rent.de - - [01/Aug/1998:00:00:02 +0100] "GET /doc.html HTTP/1.1" 200 393
```

Combined Logfile Format (DLF)

Some server use the so-called *Combined Logfile Format* to add the referrer URL and user-agent (browser) identification to the logfile entries. It looks like the CLF format followed by the referrer URL and the user-agent, where the latter two fields are surrounded by double quotes:

```
CLF "referrer_URL" "user_agent"
```

This is an example for an entry in *Combined Logfile Format* (wrapped on two lines here for readability only):

```
car.4rent.de - - [01/Aug/1998:00:00:02 +0100] "GET /doc.html HTTP/1.1" 200 393
"http://inet-tv.net/hot.html" "Mozilla/4.05 (X11; I; IRIX64 6.4 IP30)"
```

Unfortunately, the double quotes sometimes appear in broken referrer URLs, as for example in:

```
"http://www.some.host/wiredlink.html TARGET=newwin""
```

Sometimes there are even referrer URLs which contain double quotes followed by blanks, which make such entries not parseable in an unambiguous way. Although **http-analyze** recognizes the *Combined Logfile Format* automatically, and tries to do it's best to parse the referrer URL correctly, the following format, which avoids this ambiguity, should be preferred if possible.

Extended Logfile Format (ELF)

The *Extended Logfile Format* contains also the user-agent and the referrer URL as in the *Combined Logfile Format*, but in the opposite order and without the surrounding double quotes:

```
CLF user_agent referrer_URL
```

If this *Extended Logfile Format* is used, **http-analyze** searches backwards for the protocol specification of the referrer URL (to be precise, it looks for the colon in **http:**) and then for the preceeding blank. This way, even broken referrer URLs which contain blanks are handled correctly. To select this format, just edit the configuration file of your web server and select the *ELF* order of the user-agent and referrer URL fields. This is an example for an entry in *Extended Logfile Format* (wrapped on two lines here for readability only):

```
car.4rent.de - - [01/Aug/1998:00:00:02 +0100] "GET /doc.html HTTP/1.1" 200 393
Mozilla/4.05 (X11; I; IRIX64 6.4 IP30) http://inet-tv.net/index.html
```

STATISTICS REPORT

Depending on the operation mode, there are two reports: a full statistics report and a short statistics report, which might be updated more frequently. While the full statistics report contains much more details, the short statistics report covers only the most important values.

Full statistics mode

By default, **http-analyze** runs in full statistics mode. Due to technical reasons, a full statistics report will not be created before the second day of a new month, although the totals for the first day of the new month on the summary main page of the report will be updated. A full statistics report contains a detailed summary including (see the section *Interpretation of the results* for an explanation of the terms):

- the number of hits, files, pageviews, sessions, and data sent by year, month, and day
- the total amount of data requested, transferred, and saved by cache
- the total number of unique URLs, sites, sessions, agents, and referrers
- the total number of all response codes other than 200 (*OK*)
- the total number of reuqests which required authentication
- the average load per week, day, hour, minute and second
- the top 7 days, 24 hours, 5 minutes and 5 seconds
- the top 30 most commonly accessed URLs (hits, pageviews, sessions, data sent)
- the 10 least frequently accessed URLs (hits, pageviews, sessions, data sent)
- the top 30 client domains, browser types, and referrer hosts
- the overview/detailed list of all files, sitenames, browser types, and referrer URLs
- the list of all Code 404 (*Not Found*) responses

Short statistics mode

In short statistics mode, **http-analyze** creates a short summary including only the number of hits, files, pageviews, sessions, and the amount of data sent per day. Since the short statistics report does not contain as many details as a full statistics report, it requires only a fraction of processing time to create it.

A short statistics report is created if requested explicitely and also in full statistics mode for the current month. This way, on the first day of a new month, when no full statistics can be generated due to technical reasons, a short statistics report is available at least.

Running **http-analyze** in short statistics mode explicitely may be useful if the load on the server increases when creating full reports very frequently. For example, a short statistics report can be generated twice per hour, while a full statistics report is created only twice per day.

USER INTERFACES

There are two user interfaces to the statistics report: a conventional interface suitable for any browser and a frames-based interface which requires JavaScript.

The conventional interface

The conventional interface appears as in version 1.9e if JavaScript is disabled in your browser or the option **–g** was specified at invocation of **http-analyze**. If JavaScript is enabled, the following separate windows are used for different parts of the report to allow for easy navigation:

The Main window
: This window is used for most parts of the report such as the yearly, monthly, daily and weekly summaries, the *Top N* lists and the overviews. Hotlinks in the *Top N* most often point to the corresponding page, which is then displayed in the *Viewer window* if the link is followed, while hotlinks in the overviews point to the detailed lists, which show up in the *List window*.

The Navigation window
: If JavaScript is enabled in your browser and a summary for a year or a month is loaded in the main window, a small window containing a navigation panel will pop up. If JavaScript is disabled, the navigation links appear at the bottom of the monthly summary pages. In this case, use the *Back* button of your browser for navigation.

The List window
: This window is used for the detailed lists of URLs, sites, browser types and referrer URLs. A separate window for those (often large) lists causes them to be loaded only once if the links in the *Main window* are followed and the *List window* is still open.

The Viewer window
: This window is used for external pages which are loaded by following hotlinks in the statistics report. This way, you can visit the pages referred to in the report without having to go forth and back between the report and the pages listed there.

The 3D window
: This window is used for the 3D (VRML) model of the statistics. If you have JavaScript enabled, the window's size will be set to the smallest possible size so that the 3D model fits onto the screen or to the dimensions given in the **3DWinSize** directive.

Conventional Interface (JavaScript-enabled)

The frames-based interface

The frames-based interface requires a JavaScript-enabled browser. It contains the following frames and windows:

The Navigation frame

This frame contains navigation buttons and text. You can specify it's width using the **NavigFrame** directive in the configuration file.

The Main frame

This frame is used for most parts of the report such as the yearly, monthly, daily and weekly summaries, the *Top N* lists and the overviews. Hotlinks in the *Top N* lists point most often to the corresponding page, which is displayed in the *Viewer window* if the link is followed, while hotlinks in the overviews point to the detailed lists, which show up in the *List window*.

The List window

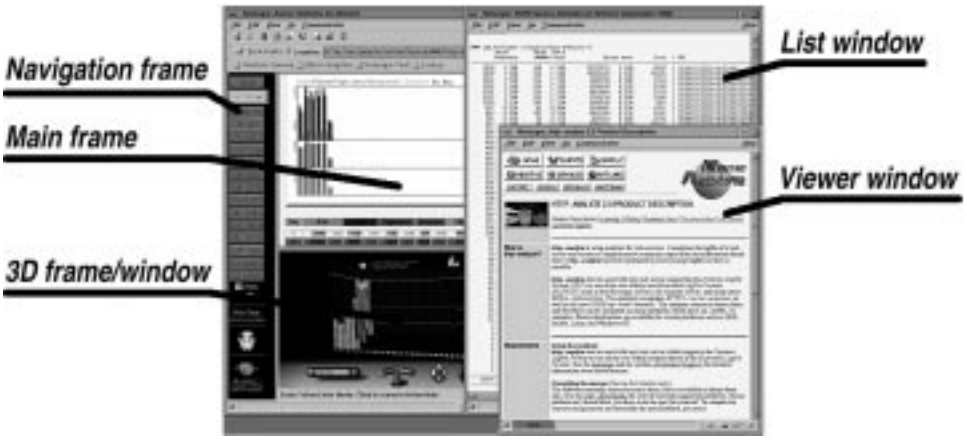
This (separate) window is used for the detailed lists of URLs, sites, browser types and referrer URLs. A separate window for those (often large) lists causes them to be loaded only once if the links in the *Main window* are followed and the *List window* is still open.

The Viewer window

This (separate) window is used for external pages which are loaded by following the hotlinks in the statistics report. This way, you can visit the pages referred to in the report without having to go forth and back between the report and the pages listed there.

The 3D window

This window is used for the 3D (VRML) model of the statistics. Depending on the setting of the **3DWindow** directive in the configuration file, this is either a separate window (*external*) or a new frame (*internal*) inside the *Main frame* (actually, two frames are created which replace the former *Main frame* when the 3D model is being displayed). In case of a separate (external) *3D window*, you can specify it's dimensions using the **3DWinSize** directive.

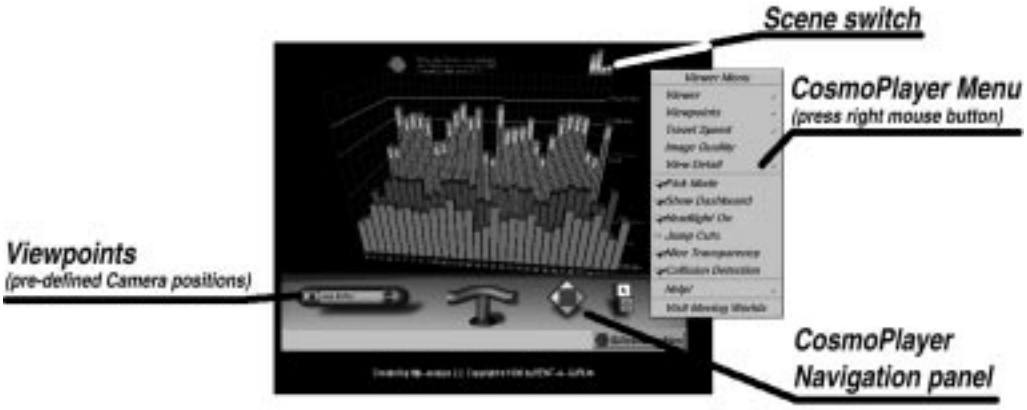


Frames-based interface

The 3D model

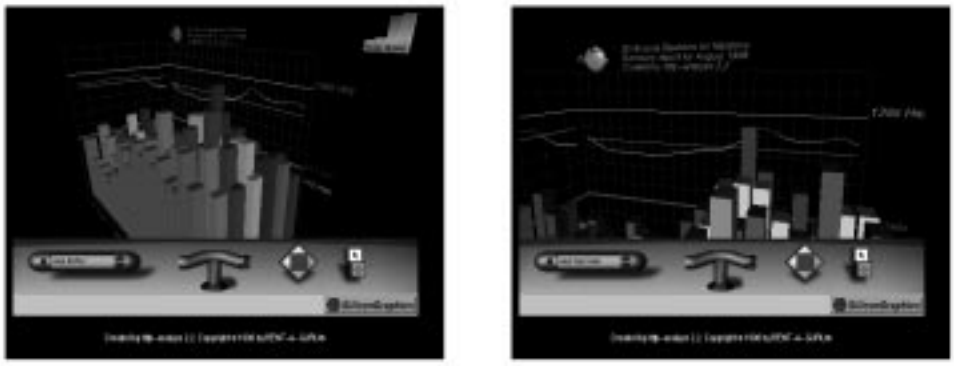
The 3D model requires a VRML 2.0 plug-in such as CosmoPlayer from Cosmo Software (http://cosmo.sgi.com/). Using this plug-in, which is available for Netscape on Silicon Graphics systems and Netscape/MSIE on Windows NT, you can "walk" or "fly" through the model and view the scene from all sides. And if you look at the models, don't forget to touch the buddha appearing in our 3D logo on top of the statistics report in the yearly summary pages!

The 3D model contains two *scenes* (statistics models): one scene showing the hits, 304's, sites and data sent by day and another scene showing the server's load by weekday and hour. To view the second scene, click on the *scene switch* on the right top of the model. To navigate through the 3D space, use the *Viewpoints* and the CosmoPlayer *Navigation panel*. For customization of CosmoPlayer use the pop-up menu, which appears if you press the right-most mouse button.



The 3D model (first scene)

The 3D representation of hits by weekday and hour in the second scene allow easy identification of the time your server has been most busy serving requests. In the figure below, most hits did occur Friday between 16:00 and 17:00.



The 3D model (second scene)

INTERPRETATION OF THE RESULTS

http-analyze shows you a summary of the content of your server's logfile. It collects information from the logfile entries, sets them into some relationship and creates a summary as a result of this analysis. The following is an explanation of the terms used in the report:

Hits (color key: green) A hit is any response from the server on behalf of a request sent from a browser. This includes **any** response from the server, not only text files or documents. For example, if a HTML page is requested, which has two inline images, the server would generate three hits: one hit for the HTML page itself and two hits for the inline images. On the other side, if an invalid URL is requested, the server would respond with a Code 404 (*Not Found*) status code, which also generates a hit.

Files (color key: blue) If the user requests a document and the server successfully sends back a file for this request, this is counted as a Code 200 (*OK*) response. Any such response is counted for as a file. Again, "file" here means any kind of a file, no matter whether it contains text (documents, directory listings) or binary data (images, applets, etc.).

Code 304 (Not Modified)

(color key: yellow) A Code 304 (*Not Modified*) response is generated by the server if a document hasn't changed since the last time it was transferred to some site.

If a browser has access to a local copy of a document requested by the user – either through it's local disk cache or through a caching server on the way between the browser and the web server –, it sends out a conditional request, which contains the modification date of the document as stored in the browser's or the caching server's local cache. If the document has changed since then, the server re-transmits the new document. If it hasn't changed meanwhile, the server sends back a Code 304 response and the browser uses it's local copy.

While this technique can significantly reduce network traffic, it causes an inaccuracy in the statistics report regarding the number a document is actually transmitted to some visitor because of two reasons: First, the browser usually sends only one such a conditional request per session if it still holds an up-to-date copy of the file. Second, caching servers often serve many thousands of users. So if you see some requests from a caching server of an online service for example, this could be caused by thousands of users requesting a certain document or just one person with a browser configured to not cache anything at all. However, the ratio between "files" and "304's" reflects the efficiency of overall caching mechanisms for at least those hits which made their way to the server.

Pageviews

(color key: magenta) The analyzer classifies all URLs which match certain patterns as pageviews (text files). Patterns may be defined using an option or a directive in the configuration file. The analyzer automatically pre-defines the suffix **.html** as a pageview. Classifying requests of certain files as pageviews allows you to estimate the number of "real" documents transmitted by your server. If used correctly, **http-analyze** rates text files (documents) as pageviews, which do not include images, CGI scripts, Java applets or any other HTML objects.

KBytes transferred

(color key: orange) This is the amount of data sent during the whole summary period as reported by the server. Note that some servers do log the size of a document instead of the actual number of bytes transferred. While in most cases this is the same, if a user interrupts the transmission by pressing the browser's stop button before the page has been received completely, some servers (for example all Netscape web servers) do not log the amount of data transferred but the amount of data which *would* have been transferred if the user would have completely loaded the page.

KBytes requested

This is the amount of data requested during the whole summary period. **http-analyze** computes this number by summing up the values of *KBytes transferred* and *KBytes saved by cache* (see below).

KBytes saved by cache

The amount of data saved by various caching mechanisms. This value is computed by multiplying the number of Code 304 (*Not Modified*) requests per file with the size of the corresponding file. Because **http-analyze** can determine the size of a file only if the file has been requested at least once in the same summary period, the values for *KBytes saved by cache* and *KBytes requested* are just approximations of the actual values.

Unique URLs

Unique URLs are the number of all different, valid URLs requested in a given summary period. This shows you the number of all different files on your web server requested at least once in the corresponding summary period.

Referrer URLs

If a document on your server is requested because a hypertext link to it on a page of a foreign web server is followed, the name of this server gets logged as the *referrer URL* (the URL of the page referring to your document). Note, that if the URL is specified manually in the browser, no referrer URL gets logged. Such requests are collected under *Unknown* in the referrer URL part of the report.

Self-referrer URLs

If a document loaded by the browser contains any inline objects (images, applets, etc.) on the same server, they are requested for in separate requests. Those requests are so-called self-referrers, because they have the own hostname in the referrer URL. If configured correctly, **http-analyze** separates all self-referrer URLs from the rest of the external referrer URLs in the statistics report.

Unique sites

This is the number of all unique hosts which did access the server during the period of the statistics report. Each different host gets counted only once per period, so this number tells you how many sites did request documents from your server per month.

Sessions

(color key: red) Similar to unique sites, this is the number of unique hosts which did access the server during a given *time-window*, which defaults to one day for backward compatibility. This number therefore reflects the different sites per day if the time-window hasn't be changed with the option **–u** or the **Session** directive in the configuration file. You can increase or decrease the time-window used to calculate sessions. For example, if you set a time-window of two hours, all accesses from the same host in less than 2 hours after it's first access are lumped together into one session. Any access more than 2 hours later will be counted as a new session.

Request Method

The browser uses a certain method to request a document from a web server. For example, documents, images, applets, etc. are usually requested using the **GET** method. Other often used methods are the **HEAD** method to request information about a document such as it's size without have the server send it's actual content, and the **POST** method, a special way to transfer user input from forms into CGI scripts.

Although all logfile entries with a valid request method are accounted for as hits, only URLs requested using either the **GET** or the **POST** method are processed further. The remaining hits are summarized under *Request Methods other than GET/POST*.

Response Codes

In reply of a request from a browser, the server sends back a status code such as a Code 200 (*OK*) or Code 404 (*Not Found*) response. Similar to the request methods, the analyzer will account any valid response code as a hit, but it will only process those URLs, which did cause a Code 200 (*OK*), Code 304 (*Not Modified*), or Code 404 (*Not Found*) response from the server. All other responses are summarized in the monthly summary page under *Other Response Codes*. See the HTML specification at *http://www.w3.org/* for information about all valid response codes. **http-analyze** recognizes HTTP/1.1 responses according to RFC 2068.

What the report does not show ...

Due to the nature of the HTTP protocol used for communication between the browser and the server and due to the type of information available in the server's logfile, the analyzer can **not**:

- identify a person as a visitor of your server,
- count the number of visitors of your server,
- track the way a visitor takes through your site,
- measure the time a visitor sees a page of your server,
- inform you about the sudden death of the visitor while looking at your homepage,
- nor show any other information not in the server's logfile.

Even if you classify certain URLs as pageviews or use a specific time-window to count sessions, this does in no way tell you anything about the number of visitors of your server.

However, if you use an appropriate server structure with files grouped by type or if you use the **HideURL** directive to group unstructered files together, the statistics report can show a trend or a tendency. Following the numbers for some time, you soon get a feeling which documents are most interesting for the visitors of your site.

OUTPUT FILES

A statistics report is created in the output directory specified at invocation of **http-analyze** or in the current directory if no output directory is given. Starting with version 2.0, all output files are placed into separate subdirectories to reduce the number of directory entries per report. Those subdirectories are named **www***YYYY*, where *YYYY* is the year of the period covered by the report. This ensures century compliance for the latest version of **http-analyze** and also makes older (non-compliant) files from the 1.9e version fully Year 2000 compliant without having to re-generate the old statistics. Of course, all HTML output files created by **http-analyze** are HTML 3.2 compliant and have been validated using *weblint*.

The analyzer can be instructed to place files with "private" data such as overviews and detailed lists of files, hosts, browser types, and referrer URLs in a separate ("private") subdirectory. The web server then can be configured to request authentication for access of files in this directory (see the option **–p** and the **PrivateDir** directive in the configuration file). **Note:** for protection of the whole report, you would configure your web server to request authentication for any file in the statistics output directory. A "private" area is needed only if you want to secure certain lists, while granting access to the rest of the statistics report.

The following list shows all files created for a full statistics report:

index.html

is the main page for a given year and contains the total number of *hits*, *files*, *pageviews*, *sessions* and *data sent* per month in tabular and graphical form for the last 12 months. At the end of the year, this file reflects the values for the whole year, while the values for the last 12 months will be written into another index file in a new directory **www***YYYY*. This page is displayed in the *Main window*.

stats*MMYY***.html** and **totals***MMYY***.html**
contain the total summary for the month *MM* of year *YY* in tabular form. The file **totals***MMYY***.html** is the frames version of the report in **stats***MMYY***.html**. In the conventional interface, this page is displayed in the *Main window*.

jsnav.html and **nav***MMYY***.html**
Navigation panels for JavaScript-enabled browsers, shown in the *Navigation window*.

days*MMYY***.html**
contains the number of hits, files, pageviews, sessions and data sent per day for the month *MM* of year *YY*. This report is displayed in the *Main window*.

avload*MMYY***.html**
contains a graphical representation of the average hits per weekday/hour and the top seconds, minutes, hours, and days of the current period. Appears in the *Main window*.

country*MMYY***.html**
contains the list of all countries the visitors of your web server came from. This information is determined by analyzing the *top-level domain (TLD)* of the hostname assigned to a system in the *Domain Name System (DNS)*. The country report is displayed in the *Main window*.

Note 1: The country list is meaningful only for ISO two-letter domains. All other domains (**.com**, **.org**, **.net**, etc.) are used by organizations world-wide, so they are not assigned a country, but listed literally in the charts. The ISO country code for the U.S. is **.us**, by the way ...

Note 2: If DNS lookups are disabled in your web server or if the system accessing your server has not been assigned a symbolic hostname in the *Domain Name System* for whatever reason, **http-analyze** can not determine the country (domain) a system is located in. All hosts without a hostname registered in the DNS will show up as *Unresolved* in the country list. Since some systems are intentionally not registered in the DNS, a percentage of up to 40% for unresolved IP numbers is absolutely normal.

3Dstats*MMYY***.html**, **3Dstats***MMYY***.wrl.gz**, **3Dstats***YYYY***.html**, **3Dstats***YYYY***.wrl.gz**
are pre-requisite files for the 3D statistics models in the *Virtual Reality Modeling Language (VRML)*. Those models are created if the option **–3** is given at invocation of **http-analyze**. To view those models, you need a VRML 2.0 compatible plug-in such as the free *CosmoPlayer* from Cosmo Software, which is currently available for Netscape Communicator and MS Internet Explorer. See *http://cosmo.sgi.com/* for more information about Cosmo Software. All 3D models are displayed in the *3D window*, so that you can compare them against the graphs in the conventional report.

While the monthly models may be displayed separately on any system with a VRML-compliant browser, the yearly model (with all other twelve monthly models embedded in it) is suitable only on a fast graphics workstation due to it's increased complexity. Therefore, if only **–3** is given, the yearly model is replaced by a logo which can be displayed again on any system.

In case you have a workstation available for display of the model, you can generate a world with all twelve monthly models embedded in it by specifying a prolog file using the option **–P** or the **VRMLProlog** directive in the configuration file (the file **3Dprolog.wrl** is provided as an example). The report then will include a button to choose between the workstation ("SGI") and the PC version of the yearly model.

topurl*MMYY***.html**, **topdom***MMYY***.html**, **topuag***MMYY***.html**, **topref***MMYY***.html**

Those files contain the *Top Ten* lists (actually it's *Top N*, where *N* is a configurable number) of the files, sites, browser types and referrer URLs. The URLs shown in **topurl***MMYY***.html** are either the real URLs requested by the visitor or an *item* (arbitrary text) you choosed to collect certain file names under (see the **HideURL** directive in the configuration file).

The domain names shown in **topdom***MMYY***.html** are either the second-level domains of the hosts accessing your server if the DNS name is available or an item you choosed to collect certain hostnames under (see the **HideSys** directive in the configuration file). Unresolved IP numbers show up as *Unresolved*.

The file **topuag***MMYY***.html** contains a list of all different browser types *(user agents)* which have been used by visitors to access your web site. The browser type is an identification string sent by the browser and logged by the web server. Although the format for this identification string is well-defined, it isn't obeyed by any browser. If possible, **http-analyze** reduces the name of the browser in the Top lists to the browser model including the first digit of it's version number. If it is not possible to determine this information, the full name as sent by the browser is used.

The referrer URLs are the URLs of those web pages, which have a link to a page on your server, and which have been visited by the user just before following the link. Note that if the user did address a document on your server manually in his browser, no referrer URL gets logged. The browser can also choose to not send a referrer URL at all. Entries without a referrer URL appear as *Unknown* in the report. The list of referrer URLs is displayed in the *Main window*.

files*MMYY***.html**, **sites***MMYY***.html**, **agents***MMYY***.html**, **refers***MMYY***.html**

Those files contain a complete overview of the files, sites, browser types and referrer URLs, similar to the Top N lists.

lfiles*MMYY***.html**, **lsites***MMYY***.html**, **lagents***MMYY***.html**, **lrefers***MMYY***.html**

Those files contain the detailed lists of all files, sites, browser types and referrer URLs, similar to the previous lists, but sorted by item (if any) and hits. On frequently accessed sites those lists can become rather large, so they are shown in the separate *List window*.

rfiles*MMYY***.html**

contains all invalid URLs which caused the server to respond with a *Code 404 (Not found)* status. If there are large number of hits for certain files the server couldn't find, it's probably due to missing inline images or other HTML objects embedded in other pages. This report is displayed in the *Main window*.

rsites*MMYY***.html**

contains the list of reverse domains. This report is displayed in the *Main window*.

frames.html, **header.html**

This two files are required for the frames-based user interface. All other files are shared with the ones for the non-frames UI. In the frames-based UI, the *Main* window is inside the frame, while the *List* window is still an external window. The *3D window* may be inside the frame or an external window (see the **3DWindow** directive).

gr-icon.gif

This is a small icon displayed on the main page under the base directory for the statistics report (option **–o** or the **OutputDir** directive in the configuration file).

OPTIONS

–h print a short help list explaining the usage of the options. Use **–hh** to print an even more detailed help.

–d *(daily mode)* generate a short statistics report for the current month only. If a history file exists, the values for the previous days will be read from this history file and the corresponding logfile entries are skipped. If the history file does not exist, the whole logfile will be processed and a history file will be created (unless **–n** is also given).

–m *(monthly mode)* generate a full statistics report for a whole month. In this mode, the values from the history file for previous month are used to create a summary page for the last 12 months. However, the logfile entries feed into the analyzer always take precedence over the records in the history unless the option **–e** is given.

–V *(version)* print the version of **http-analyze** and exit immediately.

–3 create a 3D (VRML) model of the statistics in addition to the regular statistics report. You need a VRML 2.0 compliant plug-in such as *CosmoPlayer* from Cosmo Software to view the model.

–a ignore all URLs which required authentication. If your statistics report is available to the public, you probably do not want to have those secret URLs listed in the report. See also the **AuthURL** directive in the configuration file.

–e use the history file even in full statistics mode. If this option is given and you analyze the logfiles for several months at once (either in different files or in one single logfile), **http-analyze** uses the values recorded in the history file for previous months and skips all logfile entries up to the first day of a month not recorded in the history (usually the current month). This option is useful if, for example, you rotate your logfile once per quarter and want to have the analyzer skip all entries for a previous month which already has been processed before.

–f create also a frames-based user interface for the statistics report (requires JavaScript).

–g *(generic interface)* create a conventional (non-frames) user interface for the statistics report without the JavaScript-based navigation window.

–n *(no update)* do not update the history file. Useful to generate statistics for previous months (before the last month) without overwriting the current state of the history. Since the history is used to create the report for the last 12 months, this option must be used to not mess up the actual statistics report when analyzing an older period.

–q do not strip arguments to CGI scripts in URLs. By default, **http-analyze** strips arguments to CGI scripts from their URLs to be able to lump them together. If your server creates HTML files dynamically through a CGI script, they are reduced to the URL of the script. The option **–q** causes the analyzer to leave those argument lists intact. This way, CGI URLs with different arguments are treated as different URLs. Note that this only works for requests passing arguments using the **GET** method (see the section *Interpretation of the results* for an explanation of the request methods and the **StripCGI** directive in the configuration file).

–v (verbose) comment ongoing processing. Warnings are printed only in verbose mode. Use this option to see how **http-analyze** processes the logfile. If **–v** is doubled, the analyzer prints a dot for each new day discovered in the logfile.

–x list each image URL literally rather than lumping them together under the item "All images". Without this option, **http-analyze** comprises all images *(*.gif, *.jpg, *.ief, *.pcd, *.rgb, *.xbm, *.xpm, *.xwd, *.tif)* under the item "All images" to avoid cluttering up the lists with lots of image URLs. If **–x** is given, each image URL is listed literally unless matched by an explicit **HideURL** directive in the configuration file.

–c *cfgfile*
use *cfgfile* as the configuration file. A configuration file allows you to define the behaviour of **http-analyze** and to define the ;look & feel+ of the statistics report. See the section *Configuration File* for a description of possible settings, which are called *directives* in the following text.

–l *libdir*
use *libdir* as the central library directory where **http-analyze** looks for the pre-requisite files, buttons, and license information (usually */usr/local/lib/http-analyze*).

–o *outdir*
use *outdir* to create the statistics report in. If no output directory is given, the report is created in the current directory. See also the **OutputDir** directive.

–p *privdir*
place the detailed list of files, sites, browsers and referrer URLs into the subdirectory *privdir*. Because *privdir* is created directly under the output directory specified with **–o**, it's name may not contain any slashes ('/'). This option is useful to restrict free access to only certain parts of the statistics report. See also the **PrivateDir** directive.

–F *format*
use this logfile format. Valid values for *format* are **auto** for auto-sensing the logfile format, **clf** for the *Common Logfile Format*, or **dlf** and **elf** for the two supported forms of the *Combined/Extended Logfile Format*. See also the section *Logfile Formats* above.

–G *pattern,...*
define additional pageview patterns. All URLs matching one of the *patterns* are classified as pageviews (text files). If *pattern* starts (doesn't start) with a slash (**/**), it is treated as a prefix (suffix) each URL is compared with. The suffix **.html** is pre-defined by default. You can add 9 more patterns here, for example **.shtml**, **.text** and **/cgi-bin/**. To specify more than one suffix with a single **–G** option, use commas to separate them. See also the **PageView** directive.

–H *idxfile,...*
define additional directory index filenames. The name *index.html* is pre-defined by default. **http-analyze** truncates URLs containing an index filename so that they merge with '/' (their "base URL"). For example, */dir/index.html* is truncated to */dir/*. You can add up to 9 more names for directory index files, for example *Welcome.html* or *home.html*. See also the **IndexFiles** directive.

–I *date* skip all logfile entries until this day (exclusive). The date may be specified as *DD/MM/YYYY or MM/YYYY* , where *MM* is the number or the name of a month. Note that in full statistics mode, *DD* defaults to the first day of the month if absent. If you specify any other day in this mode, unpredictable results may occur. For example, `–I Feb` restricts the analysis to the February of the current year.

–E *date* skip all logfile entries starting from this day on (inclusive). The date format is the same as in **–I**. To restrict analysis to a certain period, specify the starting date using **–I** and the first date to be ignored using **–E**. For example, `–I Jan/98 –E Feb/98` restricts the analysis to January 1998.

–O *virtname,...*
define additional ("virtual") names for this server to be classified as *self-referrer URLs*. The server's primary name (from **-S** or **-U**) is pre-defined already. If *virtname* doesn't include a protocol spcifier, two URLs with the `http` and the `https` protocol specifier are added for each name. See also the **VirtualNames** directive.

–P *prolog*

use *prolog* as the prolog file for a yearly VRML model (optional). The file **3Dprolog.wrl** is included in the distribution as an example. Note that the resulting VRML model for a whole year is suitable only for viewing on a graphic workstation. The monthly VRML models do not need a prolog file and can be viewed on any platform without problems. See also the **VRMLProlog** directive.

–R *docroot*

restrict logfile analysis to the given Document Root. If *docroot* is prefixed by a '!', analysis takes place for all directories except *docroot*. If *docroot* does not start with a slash ('/'), it is interpreted as the name of a virtual server, which is matched against the (normally unused) second field of a logfile entry. Intented for use with (software-) virtual servers with a separate Document Root or for which the hostname is recorded in the second field of a logfile entry. See also the **DocRoot** directive in the configuration file.

–S *srvname*

use *srvname* for the server name. If no server name is defined, **http-analyze** uses the hostnamename of the system. The server name must be a full qualified domain name, not an URL. See also the **ServerName** directive.

–T *TLDfile*

use *TLDfile* for the list of valid top-level domains (TLDs). This list currently includes all ISO two-letter country domains, the well-known domains **.net**, **.int**, **.org**, **.com**, **.edu**, **.gov**, **.mil**, **.arpa**, **.nato**, and the new *CORE* top-level domains **.firm**, **.info**, **.shop**, **.arts**, **.web**, **.rec**, and **.nom**. The length of a top-level domain in the TLD file may not exceed 6 characters. If no TLD file is given, **http-analyze** uses it's built-in defaults. See also the **TLDFile** directive and the sample file **TLD** included in the distribution.

–U *srvurl*

define *srvurl* as the server URL which should be used as a prefix for the hotlinks in the URL list. Useful if the statistics report is created on a different system than the server is running on and for virtual hosts. See also the **ServerURL** directive.

–W *3Dwin*

define the window for the VRML model. The keyword *3Dwin* may be either **extern** or **intern** for display of the VRML model in a new, external window or in the lower half of the main frame respectively (meaningful only in the frames-based interface).

–s *subopt,...*

suppress certain lists in the report. See also the **Suppress** directive. *subopt* may be:

AVLoad	to suppress the average load report (top seconds/minutes/hours),
URLs	to suppress the overview and list of URLs/items,
URLList	to suppress the list of URLs/items only,
Code404	to suppress the list of Code 404 (*Not Found*) responses,
Sites	to suppress the overview and list of client domains,
RSites	to suppress the overview of reverse client domains,
SiteList	to suppress the list of all client domains/hostnames,
Agents	to suppress the overview and list of browser types,
Referrer	to suppress the overview and list of referrers URLs,
Country	to suppress the list of countries,
Pageviews	to suppress pageview rating (304's are shown instead),
AuthReq	to suppress requests which required authentication,
Graphics	to suppress images such as graphs and pie charts,
Hotlinks	to suppress hotlinks in the list of all URLs,
Interpol	to suppress interpolation of values in graphs.

–t *num* define the size of certain lists. *num* is either a positive number or the value 0 to suppress the corresponding list. You specify the list by appending one of the following characters to the number shown here as '#' (note that the characters are case-sensitive):

#**U**	# is the number of entries in the Top URL list (default: 30),
#**L**	# is the number of entries in the Least URL list (default: 10).
#**S**	# is the number of entries in the Top domain list (default: 30),
#**A**	# is the number of entries in the Top agent/browser list (default: 30),
#**R**	# is the number of entries in the Top referrer URL list (default: 30),
#**d**	# is the number of entries in the Top days table (default: 7),
#**h**	# is the number of entries in the Top hours table (default: 24),
#**m**	# is the number of entries in the Top minutes table (default: 5),
#**s**	# is the number of entries in the Top seconds table (default: 5),
#**N**	# is the size of the navigation frame (default: 120 pixels)

You can specify more than one *num* with a single **–t** option by separating them with a ',' as in `–t 20U,0L,20S`. See also the **Top*** directives in the configuration file.

–u *time* define the time-window for counting *sessions*. See *Sessions* in the section *Interpretation of the results* for an explanation of this term.

–w *hits* set the noise-level to *hits*. If a noise-level is defined, all URLs, sites, agents and referrer URLs with hits below this level are collected under the item *Noise* in the Top N lists and overviews to avoid cluttering up those lists. See also the **NoiseLevel** directive.

logfile(s)

This are the name(s) of the logfile(s) to process. If more than one file is given, they are processed in the order in which their names appear on the command line. **http-analyze** checks for the existance of all files before processing them. If a '-' is specified as the filename, standard input is read. If no file is given, the analyzer either processes the default logfile specified in the configuration file or the standard input.

CONFIGURATION FILE

The option **–c** and the environment variable **HA_CONFIG** allow to define a configuration file which contains server-specific configuration settings for **http-analyze**. However, command line options always take preceedence over the definitions in this configuration file.

The configuration file contains a single directive per line. Except for **IndexFiles**, **PageView**, **AddDomain**, **VirtualNames**, **Ign***, and **Hide***, each directive may appear only once in the configuration file.

Following a directive field there are one or two value fields, which must be separated from the directive and each other by one or more tabulators. Blanks are considered a part of the string for the third field only if there is such a field. All directive names are case-insensitive. Comment lines starting with a hash character (#) are ignored.

3DWinSize *width*×*height*

Defines the size of the 3D window. Useful for Netscape Navigator 3.X, which displays scrollbars in the 3D window with standard size (520×420 pixels). Example:

```
3DWinSize       540x450
```

3DWindow *keyword*

Defines the 3D window the VRML model is displayed in (same as option **–W**). The *keyword* may be **extern** (default) or **intern** for display of the VRML model in a new, external window or in the lower half of the main frame respectively. Example:

```
3DWindow        intern
```

AddDomain *domain string*

Add entries to the domain table causing certain *domains* to be allocated to the "mock" domain *string*. Wildcards in *domain* are ignored. This directive is useful to collect certain hostnames (for example the hosts of world-wide operating online services), under some *string* (item) instead of the country they seem to originate from. Example:

```
AddDomain       .compuserve.com CompuServe
```

AuthURL *boolean value*

Defines whether URLs which required authentication are to be skipped. By default, such URLs show up in the report just like all other URLs. Setting **AuthURL** to *Off*, *No*, *None*, *False*, or *0* causes the analyzer to skip those URLs in the logfile (if your statistics report is available to others, you probably do not want to have secret URLs listed there). Example:

```
AuthURL         No
```

CustLogoW *image srvurl* and **CustLogoB** *image srvurl*

Define images for use as customer logos in the statistics report. This feature is available only in the commercial version of the analyzer. *image* is the name of the image file relative to the output directory **OutputDir** and *srvurl* is the URL to be followed if the user clicks on the image. To use your own logos create two images – one for use with a white background (**CustLogoW**) and the other one for use with a black background (**CustLogoB**). The images should be approximately 72×72 pixels in size and must be placed into the buttons subdirectory of the output directory (*OutputDir*`/btn`). Then define the appropriate **CustLogo** directives and generate a new report with your company's logo. Example:

```
CustLogoW  btn/mycompany_sw.gif   http://www.mycompany.com/
CustLogoB  btn/mycompany_sb.gif   http://www.mycompany.com/
```

DefaultMode *mode*

The default operation mode of **http-analyze**. The value field contains either the keyword **daily** for short statistics mode or **monthly** for full statistics mode (see also options **–d** and **–m**). If left undefined, the default is full statistics mode (**monthly**). Example:

```
DefaultMode     daily
```

DocRoot *docroot*

Restricts logfile analysis to the given Document Root (same as option **–R**). If *docroot* is prefixed by a '!', analysis takes place for all directories except *docroot*. If *docroot* does not start with a slash ('/'), it is interpreted as the name of a virtual server, which is matched against the (normally unused) second field of a logfile entry. Intented for use with (software-) virtual servers with a separate Document Root or for which the hostname is recorded in the second field of a logfile entry. Example:

```
DocRoot         /customer/
DocRoot         www.customer.com
```

HTMLPrefix *prefix* and **HTMLTrailer** *trailer*

The HTML *prefix* and *trailer* to be printed after the header section and at the end of the page. If defined, the **HTMLPrefix** string must include the <BODY> tag. If a *filename* is given instead of the *prefix* or *trailer*, the HTML code is taken from this file. Example:

```
HTMLPrefix      <BODY BGCOLOR="#FF0000">
HTMLTrailer     <A HREF="/intern/">Back</A> to the internal page.
```

HeadSize *size*, **TextSize** *size*, **SmallSize** *size* and **ListSize** *size*

The font sizes for headings (navigator default, usually 3), regular text (default: 2), small text (default: 1) and lists (default: 2). **TextSize** replaces the former **FontSize**, which is still recognized. Example:

```
HeadSize        4
TextSize        3
SmallSize       2
```

HeadFont *fontlist*, **TextFont** *fontlist* and **ListFont** *fontlist*

The fonts to use for headers, for regular text, and for the detailed lists. If unset, the analyzer uses a list of common serif-less fonts for headers and regular text and a monospaced (fixed) font for the detailed lists. To force the navigator's default for fonts, use the keyword **default** as the fontname. Example:

```
HeadFont        Helvetica,Arial,Geneva,sans-serif
TextFont        Helvetica,Arial,Geneva,sans-serif
ListFont        Courier,fixed
```

HideAgent *agent string*

Hide certain browsers under an arbitrary *string* (item). Needed only for a certain browser whose vendor still can't spell it's name correctly. Only the leading part of the browser type is compared against *agent*, so no wildcards are needed in the second field. Example:

```
HideAgent       Mozilla/4.0 (compatible; MSIE 4.   MSIE 4.*
HideAgent       Mozilla/3.0 (compatible; MSIE 3.   MSIE 3.*
```

HideRefer *referrer string*

Hide certain referrer URLs under an arbitrary *string* (item). Useful to map different referrer URLs for a given host to a common name. Since only the leading string of the referrer URL is compared against *referrer*, there is no need to specify wildcards. As in **HideAgent**, a wildcard suffix is removed from the string, while a wildcard prefix is taken literal.

If the second argument contains a string in square brackets, this defines the CGI parameter which specifies the search key for search engines. In this case, the search key will be extracted from the argument list and prominently displayed after the name of the search engine/web server. See also the file **sample.conf** included in the distribution for more examples on how to use the **HideRefer** directive. Example:

```
HideRefer       http://altavista.digital.com/   AltaVista [q=]
HideRefer       http://lycospro.lycos.com/      Lycos [query=]
HideRefer       http://www.excite.com/          Excite [search=]
HideRefer       http://www.dino-online.de/      Dino Online [query=]
```

HideSys *hostname string*

Hide a *hostname* under an arbitrary *string* (item). The string may contain blanks. If the first character of *string* is a '[', this item is suppressed in the *Top N* lists. Hidden items are accounted for separately, but in the summary they are collected under the description defined with this directive. You may use the wildcard character '*' as either a prefix or as a suffix of the *hostname* (as in ***.host.com** and **192.168.12.***), bot not as both. Hostnames are case-insensitive. When building the list of countries, **http-analyze** determines the country from the top-level domain given in *hostname*. If *hostname* is an IP number, you can optionally define the top-level domain it should be accounted for by appending the domain in square brackets to the *string* as shown below. Example:

```
HideSys         *.mycompany.com MY COMPANY
HideSys         192.168.12.*    MY COMPANY [COM]
```

HideURL*url string*

Hide an *URL* under an arbitrary *string* (item). The string may contain blanks. If the first character of *string* is a '[', this item is suppressed in the *Top N* lists. Hidden items are accounted for separately, but in the summary they are collected under the description defined with this directive. You may use the wildcard character '*' as either a prefix or as a suffix of the *URL* (as in ***.map** and **/subdir/***), bot not as both. URLs are case-sensitive. Note that images are hidden automatically under *All images* unless **–x** was specified. See the **sample.conf** file included in the distribution for more examples. Example:

```
HideURL         *.map           [All image maps]
HideURL         /robots.txt     [Robot control file]
HideURL         /newsletter/*   MyCompany's Monthly Newsletter
HideURL         /~delta-t/      DELTA-t Homepage
```

IgnURL *url* and **IgnSys** *hostname*

Ignore entries with a specific URL or accesses from a certain system. You may use the wildcard character '*' as either a prefix or as a suffix of the URL or the hostname (as in ***.gif**, **/subdir/file*** and ***.host.com**), but not as both. Note that all logfile entries are compared against this list while **http-analyze** reads the logfile opposed to the **HideURL** and **HideSys** directives, which are looked up for when all entries have been reduced to the set of unique URLs and hostnames, respectively. Therefore, many **IgnURL**/**IgnSys** definitions will significantly increase processing time of **http-analyze**. Example:

```
IgnURL          *.gif,*.jpg,*.jpeg
```

IndexFiles *idxfile [,idxfile ...]*

Define additional directory index filenames (same as option **–H**). The name *index.html* is pre-defined by default. **http-analyze** truncates URLs containing an index filename so that they merge with '/' (their "base URL"). For example, */dir/index.html* is truncated to */dir/*. You can add up to 9 more names for directory index files. Note that each name requires another table lookup, which may significantly increase processing time. Example:

```
IndexFiles      Welcome.html,home.html,index.htm
```

Language *locale*

Not available yet: Use given message catalogue for the language in the statistics report. By default, the message catalogue selected by the current locale is used. This directive may be used to overwrite the locale used by **http-analyze** to find the correct message catalogue.

LogFile *filename*

The name of the server's logfile. If you define a default name for the logfile, this file is processed if no other filenames are explicitely specified on the command line. Without such a definition, **http-analyze** always reads *stdin* if no other filename is given. Example:

```
LogFile         /usr/ns-home/www/logs/access
```

LogFormat *format*

use this logfile format. Valid values for *format* are **auto** for auto-sensing the logfile format, **clf** for the *Common Logfile Format*, or **dlf** and **elf** for the two supported forms of the *Combined/Extended Logfile Format*. See the section *Logfile Formats* above for a description of the formats supported by **http-analyze**. Example:

```
LogFormat       clf
```

NavWinSize *width*×*height*

Defines the size of the navigation window which pops up in the conventional interface if JavaScript is enabled. Useful if the browser displays scrollbars when the default size of 420×190 is used. Example:

```
NavWinSize      440x200
```

NavigFrame *size*

Defines the size of the navigation frame in pixels. Useful if the browser displays scrollbars when the default size of 120 pixels is used. Example:

```
NavigFrame      140
```

NoiseLevel *hits*

set the noise-level to *hits*. If a noise-level is defined, all URLs, sites, agents and referrer URLs with hits below this level are collected under the item *Noise* in the Top N lists and overviews to avoid cluttering up those lists. Example:

```
NoiseLevel      7
```

OutputDir *directory*

The name of the directory where the output files should be created (same as option **–o**). If left undefined, output files are created in the current directory. Example:

```
OutputDir       /usr/www/htdocs/stats
```

PageView *pattern [,pattern ...]*

define additional pageview patterns (same as option **–G**). All URLs matching one of the *patterns* are classified as pageviews (text files). If *pattern* starts (doesn't start) with a slash (*/*), it is treated as a prefix (suffix) each URL is compared with. The suffix **.html** is pre-defined by default. You can add 9 more patterns here, for example **.shtml**, **.text** and **/cgi-bin/**. Note that each pattern requires another table lookup, which may significantly increase processing time. Example:

```
PageView          .shtml,.text,/cgi-bin/
```

PrivateDir *privdir*

The name of a private directory where the detailed lists of files, sites, browsers, and referrer URLs should be created (same as option **–p**). Because *privdir* is created directly under the output directory specified with **–o**, it's name may not contain any slashes ('/'). This option is useful to restrict free access to certain parts of the statistics report only: Instead of securing the whole statistics report, you can have certain lists separated from the rest of the report and then have the server request authentication for access of this lists. Example:

```
PrivateDir        lists
```

RegInfo *customer_name registration_ID*

Defines the customer's name and the registration ID, which are both shown on the main page in the summary report. Example:

```
RegInfo           MyCompany        3745JMJZ00000311300000682344
```

ReportTitle *title*

The document title to use in the statistics report. Example:

```
ReportTitle       Access Statistics for MyCompany
```

ServerName *srvname*

The official name of the server (same as option **–S**). If no server name is defined, **http-analyze** uses the hostnamename of the system. The server name must be a full qualified domain name, not an URL. Example:

```
ServerName        www.mycompany.com
```

ServerURL *srvurl*

The URL of the server to be used for hotlinks in URL lists (same as option **–U**). Useful if the report for your web server is published on another server, for example on an internal developement machine. Also necessary for (software-) virtual servers to have **http-analyze** generate correct hypertext links in the report. Example:

```
ServerURL         http://www.mycompany.com
```

Session *time*

The time-window for counting *sessions*. All unique hosts accessing your server more than once inside this time-window, are accounted for as the same session. If the distance between two adjacend accesses from the same host is greater than the time-window, the accesses from this host are accounted for as different sessions. Example:

```
Session           4 hours
```

StripCGI *boolean value*

Defines the handling of arguments to CGI scripts in URLs. By default, **http-analyze** strips arguments to CGI scripts from their URLs to be able to lump them together. If your server creates HTML files dynamically through a CGI script, they are reduced to the URL of this script. Setting **StripCGI** to *Off*, *No*, *None*, *False*, or *0* causes the analyzer to leave those argument lists intact. This way, CGI URLs with different arguments are treated as different URLs. Note that this only works for requests passing arguments using the **GET** method (see *Interpretation of the results* for information about request methods). Example:

```
StripCGI          No
```

Suppress *subopt,...*

Suppress certain lists in the report (same as **–s**). *subopt* may be one of:

AVLoad	to suppress the average load report (top seconds/minutes/hours),
URLs	to suppress the overview and list of URLs/items,
URLList	to suppress the list of URLs/items only,
Code404	to suppress the list of Code 404 (*Not Found*) responses,
Sites	to suppress the overview and list of client domains,
RSites	to suppress the overview of reverse client domains,
SiteList	to suppress the list of all client domains/hostnames,
Agents	to suppress the overview and list of browser types,
Referrer	to suppress the overview and list of referrers URLs,
Country	to suppress the list of countries,
Pageviews	to suppress pageview rating (304's are shown instead),
AuthReq	to suppress requests which required authentication,
Graphics	to suppress images such as graphs and pie charts,
Hotlinks	to suppress hotlinks in the list of all URLs,
Interpol	to suppress interpolation of values in graphs.

Example:

```
Suppress        Country,Interpol
```

TLDFile *filename*

use *filename* for the list of top-level domains (same as option **–T**). This list includes all ISO two-letter country domains, the well-known domains **.net**, **.int**, **.org**, **.com**, **.edu**, **.gov**, **.mil**, **.arpa**, **.nato**, and the new *CORE* top-level domains **.firm**, **.info**, **.shop**, **.arts**, **.web**, **.rec**, and **.nom**. The length of a domain in the TLD file may not exceed 6 characters. **http-analyze** uses it's built-in defaults, if no TLD file is given. Example:

```
TLDFile         /usr/local/lib/http-analyze/TLD
```

Top{**Days,Hours,Minutes,Seconds,URLs,Sites,Agents,Refers**}, **LeastURLs**

Defines the size of certain Top N tables and lists. If set to zero, the corresponding list will be suppressed. Example:

```
TopURLs         20
LeastURLs       0
TopDays         14
```

VirtualNames *virtname,...*

The list of additional ("virtual") names for this server to be classified as *self-referrer URLs*. The server's primary name (from **ServerName** or **ServerURL**) is pre-defined already. If *virtname* doesn't include a protocol specifier, two URLs with the `http` and the `https` protocol specifier are added for each name. Since self-referrers are suppressed from the list of referrer URLs, the remaining entries give a good impression about external pages referring to some document on your site. Example:

```
VirtualNames    www2.mycompany.com,mycompany.com
VirtualNames    www.customer.com,customer.com
VirtualNames    http://www.other.com,https://secure.other.com
```

VRMLProlog *file*

The name of a prolog file for a yearly VRML model (same as option **–P**). Pathnames not beginning with a '/' are relative to **OutputDir**. If a prolog file is given, an additional yearly model with all 12 monthly models embedded as inlines is created. This model may be displayed only on graphics workstation. See the section *Output files* for further information about this yearly model. Example:

```
VRMLProlog      3Dprolog.wrl
```

EXAMPLES

After successful compilation of **http-analyze** you can create a statistics report before you choose to install the program permanently. To do so, create a subdirectory for the output files to avoid cluttering up the directory and install the required files using the **ha-setup** utility:

```
http-analyze setup
------------------

1) Set up an analyzer configuration for a virtual web server
2) Install the required files in a statistics output directory
3) Brand your copy of http-analyze with the registration ID
4) Exit

Please select a function (1-4) [1]: 2
Install required files for http-analyze
---------------------------------------

This script copies the required files (3D*, btn/*) into the statistics
...
Name of the HTML output directory: testd
Directory testd doesn't exist, create it (y/n) [y]: <RETURN>

Now enter the name of the directory containing the required files.
...
Directory containing required files (3D*, btn/*) [files]: <RETURN>

Required files have been copied into testd
```

Then, run the analyzer on your web server's logfile. For example, if the name of the logfile is */usr/ns-home/www/logs/access*, use the following command to create a full statistics including a frames-based interface and a 3D (VRML) model in the newly created directory **testd**:

```
$ http-analyze -vm3f -o testd /usr/ns-home/www/logs/access
http-analyze 2.2 (IP22; IRIX 6.2), Copyright 1998 RENT-A-GURU(TM)
Generating full statistics in output directory 'testd'
Reading data from '/usr/ns-home/www/logs/access'
Best blocksize for I/O is 64 KB
Hmm, looks like Extended Logfile Format (ELF)
Start new period at 01/Sep/1998
Creating VRML model for September 1998
Creating full statistics for September 1998
... processing URLs
... processing hostnames
... processing user agents
... processing referrer URLs
Statistics complete until 30/Sep/1998
$
```

After the analyzer terminates, start your browser and open the file **testd/index.html**.

To permanently install the program, issue a `make install` which copies the required files in the appropriate places. To set up an analyzer configuration for a web server, choose an output directory for the statistics report and use the **ha-setup** utility to install the required files there.

Following are some more examples, which assume that the analyzer has been installed permanently. The first command processes an archived logfile *logYYYY/access.MM* from the server's log directory to create a report for January 1998 in the directory **/usr/htdocs/stats**:

```
$ cd /usr/ns-home/www/logs
$ http-analyze -vm3f -o /usr/htdocs/stats log1998/access.01
```

The next command reads the logfile entries from a pipeline and creates the statistics report for a whole year using a customized configuration file:

```
$ gzcat log1997/access.[01]?.gz |
> http-analyze -c /usr/httpd/analyze.conf -
```

REGULAR INVOCATION VIA CRON

To have statistics generated on a regular base, use the following scheme:

1) Optionally install a cron job which calls **http-analyze –d** frequently to create a short statistics report. The execution interval may range from once per day up to twice per hour depending on the size of your logfile and the time needed to analyze it. On our server, we run the daily statistics once per hour.

2) Install a cron job which calls **http-analyze –m** to create a full statistics report once per week or once per day (again depending on the size of your logfile). Note that the full statistics report is created for the first time at the second day of a new month. On our server, we create a monthly summary two times per day.

3) Create a script which rotates the server's logfile, restarts the http server, and then creates the final summary for this period. Have *cron* execute this script at 00:00 on the **first day** of a new month. See the script **rotate-httpd** for an example on how to do this for several virtual web servers running on the same machine.

4) Because of *cron*'s scheduling overhead and delays in execution of the script which rotates the logfile, heavy used servers sometimes writes a few entries for the new month in the old logfile. **http-analyze** usually detects and ignores such "white noise" at the end of a month. However, to get correct figures, in this last step you should run **http-analyze –m** on the logfile for the current month immediately after generating the statistics for the previous month.

Note that the cron jobs must run with the user ID of the owner of the directory where the HTML output files are to be created, except for **rotate-httpd**, which must run with the user ID of the server user. You should also take care to avoid running more than one **http-analyze** processes at the same time. Here are some sample *crontab*(1) entries for the scheme described above:

```
# Generate a full report twice per day at 01:17 and 13:17
17  1,13 * * *  /usr/local/bin/http-analyze -m -c /usr/httpd/analyze.conf

# Generate a short summary each hour except at 01:17 or 13:17
17  2-12 * * *  /usr/local/bin/http-analyze -d -c /usr/httpd/analyze.conf
17 14-23 * * *  /usr/local/bin/http-analyze -d -c /usr/httpd/analyze.conf

# Rotate the HTTPD logfiles at the first day of a new month at 00:00
0 0 1 * *       /usr/local/bin/rotate-httpd
```

TROUBLESHOOTING

If you discover any problems using the analyzer you may find the verbose mode helpful. Each **–v** option increases the verbosity level. In verbosity level 1, **http-analyze** comments ongoing processing; in level 2 it indicates progress by printing a dot for each new day discovered in the logfile. In level 3, a debug message for each logfile entry parsed successfully is printed and in level 4 an even more detailed message appears on standard error. Furthermore, compiling **http-analyze** without the macro *NDEBUG* includes various assertion checks in the executable.

```
$ http-analyze -vvvm3f -o testd log1998/access.08
http-analyze 2.2 (IP22; IRIX 6.2), Copyright 1998 RENT-A-GURU(TM)
Generating full statistics in output directory 'testd'
Reading data from 'log1998/access.08'
Best blocksize for I/O is 64 KB
Hmm, looks like Extended Logfile Format (ELF)
  1 01/Aug/1998:00:02:14 [262929738], req=/stats/, sz=2656 <- Code 200 OK
Start new period at 01/Aug/1998
  2 01/Aug/1998:00:02:17 [262929741], req=/logo.gif, sz=5880 <- Code 200 OK
  3 01/Aug/1998:00:02:17 [262929741], req=/btns.gif, sz=4713 <- Code 200 OK
...
```

REGISTRATION

The distribution of **http-analyze** on our web site is made available to you for evaluation purposes only. In this version an "unregistered" button will show up in the statistics report. To replace this button with the Netstore logo of the free version (for personal and educational use), just click on this "unregistered" button to follow the link to our registration form on our web site and register for a free, non-commercial version.

NON-COMMERCIAL VERSION

After registration you will receive a registration ID and two registration images as replacements for the "unregistered" buttons by email. In the private version, the Netstore logo, a Copyright note and a link to the homepage of **http-analyze** appears in the statistics report, which must be left intact according to the license, under which this software is made available to you.

COMMERCIAL VERSION

If you use **http-analyze** for commercial purposes such as providing statistics services for your customers, you must buy a *Commercial Service License* available from RENT-A-GURU® and authorized resellers. You will receive a registration ID and two registration images as replacements for the "unregistered" buttons by email from our office.

In the commercial version, the Netstore logo, the Copyright note and the link to the homepage of **http-analyze** are supressed from the statistics report (except for the logo and Copyright, which appears only once on the main page and inside the navigation frame). You can also add your Company's logo to the report using the **CustLogoW** and **CustLogoB** directives in the configuration file, which are enabled by branding the software. Except for this feature and the individual support for users of a commercial license, both versions have identical functionality.

BRANDING THE SOFTWARE

For all license types, you have to brand your copy of **http-analyze** with the registration ID and the registration images. The registration ID may be set either in a system-wide file (usually */usr/local/lib/http-analyze/REGID*) or via the **RegInfo** directives in an analyzer configuration file. The latter method requires specification of the configuration file each time **http-analyze** is invoked. If you create a system-wide registration file, you need to brand the software only once. To do so, issue the following commands as root (if you can't become root, use another directory you can write to and set the environment variable **HA_LIBDIR** to it's name):

```
# mkdir libdir
# http-analyze -r "Customer Name" regID
Registration information saved in file 'libdir/REGID'
#
```

where *libdir* is the library directory, *Customer Name* is the name of the organization this license is registered for and *regID* is the registration ID assigned to the license. Next, install the two registration images we sent you by email into the appropriate buttons subdirectory:

- If you use **http-analyze** for only one web server, copy the registration images into the buttons subdirectory (**btn**) of the corresponding statistics output directory (**OutputDir**).
- If you analyze several virtual servers on the same platform, install the registration images in the buttons subdirectory (usually */usr/local/lib/http-analyze/btn*), which should have been created during the installation process. Then, you can easily install the required files and buttons using the **ha-setup** utility by copying or linking them into the several **OutputDir** subdirectories for the virtual web servers.

After installing the buttons you have completed the registration. Now run the analyzer to create the statistics with the registered version.

YEAR 2000 COMPLIANCE

Versions 2.0 and above of **http-analyze** are fully Year 2000 compliant. There are no problems with date-related functions after the year 1999. Year 2000 compliant means, that the software does not produce errors in date-related data or calculations or experience loss of functionality as a result of the transition to the year 2000. This Year 2000 compliance statement is not a product warranty. The **http-analyze** software is provided under the terms of the license agreement included in each distribution.

DATE USAGE IN HTTP-ANALYZE

The analyzer depends on the timestamp found in the logfile of the web server. A Year 2000 compliant date format was choosen for the *Common* and *Extended Logfile Formats* from the very beginning on. This unique date format is - and ever was - required by **http-analyze** to be able to generate a statistics report, so there are no problems unless those caused by your OS (see below).

Although **http-analyze 2.X** generates two-digit years in some output filenames to retain compatibility with previous versions of the log analyzer, those files are placed in a subdirectory containing the year in four digits, which make **all** output filenames - even those generated by older versions of the log analyzer - fully Year 2000 compliant. This way, statistics reports generated by the 1.9e version of the analyzer, which originally were not Year 2000 compliant, will become compliant during the upgrade to version 2.X automatically - without re-running the statistics with the original logfiles!

The date format in the **–I** and **–E** options allows a year to be specified with two digits only. **http-analyze** interprets values greater than 70 in 1900 and values lower than 70 in 2000. This way, the analyzer covers the whole range of the time representation in modern Operating Systems. However, any other date can be specified unambiguously by using four digits for the year.

DATE USAGE IN THE OPERATING SYSTEM

Actually, there is a date-related function in modern operating systems, which may cause problems after the year 2037. For those interested in the technical details, here's why:

In operating systems the date is often represented in seconds since a certain date. For example, in Unix systems the date is represented as seconds since the birth of the OS at January, 1st 1970. This value is stored in a *signed long* (4-byte) data object, so it can represent as much as 2147483648 seconds, which equals 35791394 minutes = 596523 hours = 24855 days = 68 years. Therefore, most clocks in traditional Unix systems will overflow at January, 1st 2038 if the OS is not updated before this date.

It has been reported that a certain version of Windows doesn't recognize the Year 2000 as a leap year. Although **http-analyze** computes leap years for itself, it maps dates into weekdays using the *localtime* function, which may work correctly only if the OS itself knows about Year 2000 being a leap year. Since **http-analyze** uses several data structures depending on the operating system's idea of the time (for example, the *tm_year* variable contains the years since 1900), the software has to be updated also before the year 2038 in order to take advantage of the time representation in future OS'es.

COPYRIGHT

CREDITS

Thanks to the numeruous users of **http-analyze** for their valuable feedback. Special thanks to Lars-Owe Ivarsson forr his suggestions to optimize the parser algorithm and the code he provided as an example. Special thanks also to Thomas Boutell *(http://www.boutell.com/)* for his great GD library for fast GIF creation, without **http-analyze** couldn't produce such fancy graphics in the statistics report.

gd 1.2 is copyright 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Labs. Permission granted to copy and distribute this work provided that this notice remains intact. Credit for the library must be given to the Quest Protein Database Center, Cold Spring Harbor Labs, in all derived works.

ENVIRONMENT VARIABLES

Environment variables might work only in the Unix version of **http-analyze**.

HA_LIBDIR	name of the library directory (default: `/usr/local/lib/http-analyze`)
HA_CONFIG	name of the configuration file for **http-analyze** (no default)

FILES

This section lists all files required by **http-analyze** to create a statistics report. Those files are usually installed in the library directory as defined by the environment variable **HA_LIBDIR** or the hard-coded default (usually */usr/local/lib/http-analyze*) defined at compile-time. See also the section *Statistics Report* above for the names of the HTML output files.

btn/.gif*	Buttons and icons used in HTML output files
TLD	List of all top-level-domains
ha2.0_*.gif	**http-analyze** logos for your web site (black/white bg)
logfmt.[cde]lf	Sample logfiles in CLF, DLF and ELF format
3Dprolog.wrl	Prolog file for the yearly VRML model on SGI workstations
3DshelfMotion.wav	Sound file for the yearly model on SGI workstations
3Dlogo.wrl.gz	A stubs file for the yearly VRML model on PCs

SEE ALSO

rotate-httpd	Script to rotate the web server's logfiles
ha-setup	Script to set up the analyzer configuration for a web server
cvt_files	Script to convert older files into new 2.0 directory structure
http://www.netstore.de/Supply/http-analyze/	Homepage of **http-analyze**

NOTES

Logfile entries must be sorted in order of ascending date and time. If **http-analyze** detects logfile entries from an older month between newer ones, it prints a warning and skips all entries up to the date of the last entry processed.