NAME

http-analyze – a fast log analyzer for web servers

SYNOPSIS

http-analyze [**–{hdmBVX}**] [**–3aefgnqvxyM**] [**–b** *bufsize*] [**–c** *cfgfile*]
[**–i** *newcfg*] [**–l** *libdir*] [**–o** *outdir*] [**–p** *prvdir*] [**–s** *subopt,...*]
[**–t** *num,...*] [**–u** *time*] [**–w** *hits*] [**–F** *logfmt*] [**–L** *lang*] [**–C** *chrset*]
[**–I** *date*] [**–E** *date*] [**–G** *suffix,...*] [**–H** *idxfile,...*] [**–O** *vname,...*]
[**–P** *prolog*] [**–R** *docroot*] [**–S** *srvname*] [**–T** *TLDfile*] [**–U** *srvurl*]
[**–W** *3Dwin*] [**–Z** *showdom*] [*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. The analyzer does this by

- reading all logfiles specified on the command line,
- saving all unique (different) URLs, hostnames, referrer URLs and user agents,
- accounting for hits (successful requests), files sent, files cached, data sent, etc.,
- and finally creating a statistics report for the period detected in the logfile(s).

The resulting statistics report is a comprehensive view of the server's logfile. The server writes a logfile entry for every response on behalf of a request from a browser or a forwarding system such as proxy servers. To understand the meaning of the terms in the report, you need a little knowledge about the type of data your web server records in its logfile.

LOGFILE FORMATS

NCSA Common Logfile Format (CLF)

The basic logfile format supported by allmost all servers is the *NCSA Common Logfile Format*. It contains the following information for each request (hit):

```
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 if authentication was required.
`[date]`	The date of the access in format `[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 HTTP version number.
`clf-status`	The (numerical) response code from the server.
`ct-length`	This is either the size of the document or the data actually sent over the wire.

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

```
car.4rent.de - - [01/Aug/1999:00:00:02 +0100] "GET /doc.html HTTP/1.1" 200 393
```

W3C Extended Logfile Format (ELF)

The *W3C Extended Logfile Format (ELF)* is basically *NCSA CLF* plus user-agent and referrer URL information. **http-analyze** supports two variants of this extended format: *DLF* and *ELF*.

The *DLF* format adds the referrer URL and the user-agent in this order with or without surrounding double quotes:

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

This is an example for an entry in *DLF* format (wrapped on two lines for readability):

```
car.4rent.de - - [01/Aug/1999:00:00:02 +0100] "GET /doc.html HTTP/1.1" 200 393
"http://inet-tv.net/hot.html" "Mozilla/4.05 (X11; I; IRIX64 6.4 IP30)"
```

The *ELF* format also adds the referrer URL and the user-agent, but in the opposite order and without the double quotes:

```
CLF user_agent referrer_URL
```

This is an example for an entry in *ELF* format (wrapped on two lines for readability):

```
car.4rent.de - - [01/Aug/1999:00:00:02 +0100] "GET /doc.html HTTP/1.1" 200 393
Mozilla/4.05 (X11; I; IRIX64 6.4 IP30) http://inet-tv.net/index.html
```

The *ELF* variant is the preferred method to pass referrer URL and user-agent information. When this 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 ensures that broken referrer URLs which contain blanks or double quotes are handled correctly.

To select either logfile format, edit the configuration file of your web server and define the fields to be logged. See the web server's documentation for information how to customize logging.

Automatic detection of the logfile format

http-analyze tries to automatically detect the correct logfile format by analyzing the first few entries of a logfile (this works only if your server records a hyphen ('–') for empty referrer URL or user-agent fields). If **http-analyze** detects referrer URL and user-agent information, it assumes the *ELF* variant of the *W3C Extended Logfile Format*. To process the *DLF* variant, specify the logfile format explicitely using the option **–F**.

Logfile data used by http-analyze

The statistics report shows a summary of the information which has been recorded into the logfile by the web server. For each logfile entry **http-analyze** processes the origin (sitename) and date of the request, the request method, the URL of the requested object, the server's response on behalf of the request, the size of the requested object and optionally the user-agent and the referrer URL if sent by the client.

Note that **http-analyze** does not recognize visitors, email addresses of users visting your server, the path a user took through your web site, the last page visited by a user before leaving your site nor anything else not recorded in the server's logfile. Although hostnames are recorded for each request, they must not necessarily correspond to the real system actually used by a visitor – the request could be forwarded through a dialup service for example. Furthermore, no request may get logged by your server at all while someone is surfing through cached copies of parts of your site depending on the configuration of his/her browser ...

BASIC OPERATION

By default, **http-analyze** creates a *full statistics report* for a whole month, which contains complete details for the period determined by the timestamps of the first and last logfile entry processed. It is therefore extremly important to always feed all logfiles for a whole month into **http-analyze**, no matter how frequently you rotate (save) the logfiles.

The recommended way of providing an up-to-date statistics report for a web server is to have a script running **http-analyze** automatically on a regular base, say twice per day, and have it process the current logfile of the web server from the beginning of the current month until today. At the first of a new month, the logfile should be saved elsewhere and the web server should be restarted to create a new logfile for the new month. Then run **http-analyze** on the old (saved) logfile to create a final statistics report for the previous month. A history file is used to produce a summary for the last 12 months on the main page of the statistics report without having to analyze logfiles for those older periods again.

If you rotate the logfile more often to be able to compress them – for example, once per day –, you must uncompress and concatenate all separate logfiles for the whole month into one, chronologically ordered data stream, which the can be processed by **http-analyze**.

Full statistics report

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 the following items (see the section *Interpretation of the results* for an explanation of the terms):

- the number of hits, files sent/cached, pageviews, sessions and the amount of data sent
- the total amount of data requested, transferred, and saved by caching mechanisms
- the total number of unique URLs, sites, sessions, browser types and referrer URLs
- the total number of all response codes other than Code 200 (*OK*)
- the total number of requests 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, files, pageviews, sessions, data sent)
- the 10 least frequently accessed URLs (hits, files, pageviews, sessions, data sent)
- the Top 30 client domains, browser types, and referrer hosts
- an overview and a detailed list of all files, sitenames, browser types and referrer URLs
- a list of all Code 404 (*Not Found*) responses

Short statistics report

Since analyzing the complete logfile for a whole month increases processing time on heavily accessed web servers, you can instruct **http-analyze** to create a *short statistics report* for the current day only. In this mode, **http-analyze** updates only the daily totals for the current month in the *Hits by Day* section of the report and saves the results in a history file. If the analyzer is then run a second time to update the *short statistics report*, it skips all logfile entries from the beginning of the month until it detects any entries for the current day, which are then processed to produce an up-to-date *Hits by Day* section in the statistics report.

In *short statistics mode*, **http-analyze** needs only a fraction of processing time required for a *full statistics report*, but it updates only a very small part of the statistics report so that this should be considered an additional feature rather than a replacement for the *full statistics mode*. The recommended way for using this feature is to have **http-analyze** generate a *full statistics report* once per day or week, while generating an up-to-date *short statistics report* as often as once per hour or day.

USER INTERFACES

Two user interfaces exists for access 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 1.9 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 will 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 into 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 the latter case, use the *Back* button of your browser for navigation.

The List window

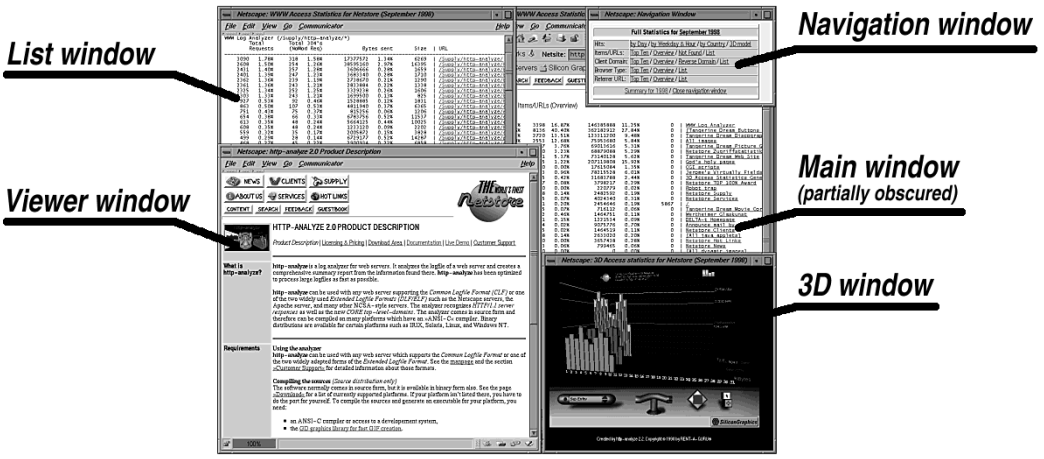
This window is used for the detailed lists of URLs, sites, browser types and referrer URLs. A separate window for those (often large) lists causes them to be loaded only once if you follow any link in the *Main window* while the *List window* is still open.

The Viewer window

This 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 itself 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 specified with 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 its 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

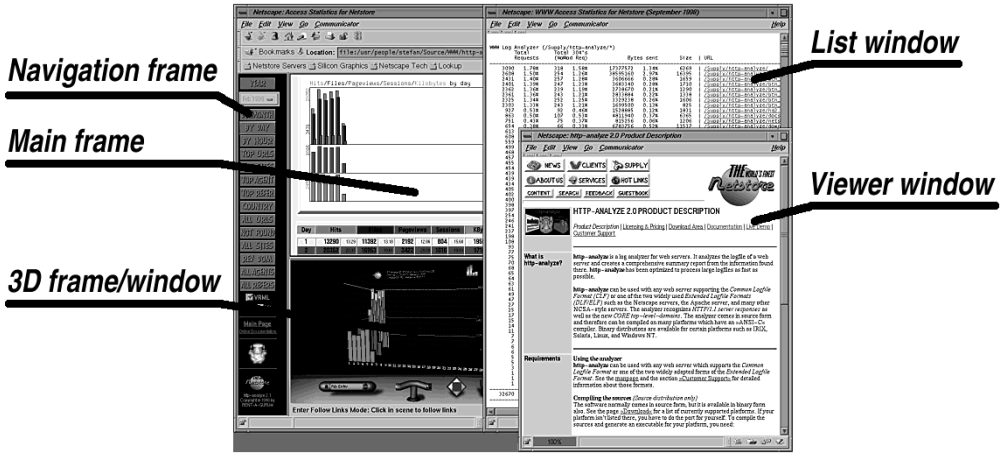
This window is used for the detailed lists of URLs, sites, browser types and referrer URLs. A separate window for those (often large) lists causes them to be loaded only once if the links in the *Main window* are followed and the *List window* is still open.

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 its dimensions using the **3DWinSize** directive.

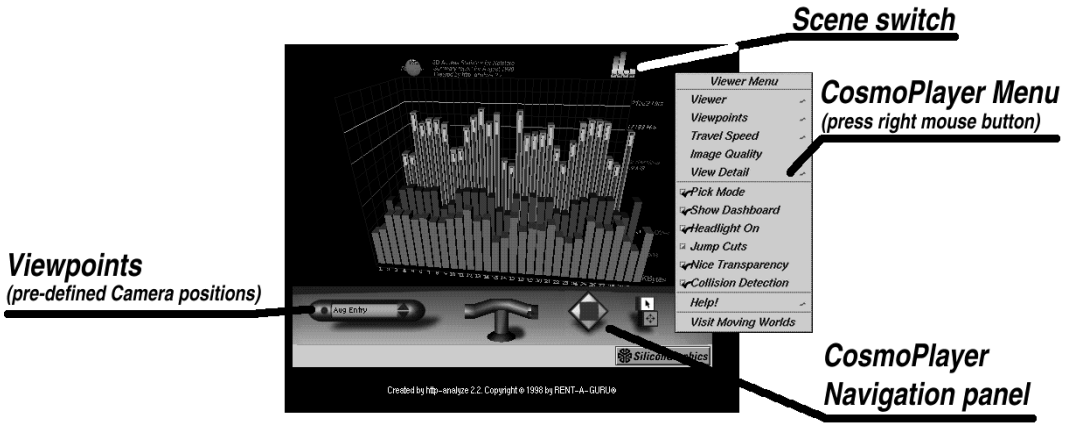


Frames-based interface

The 3D model

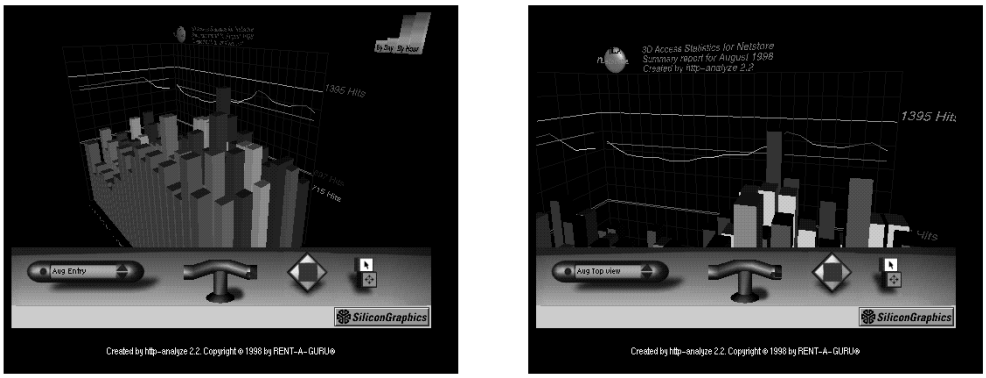
The 3D model requires a VRML 2.0 plug-in such as CosmoPlayer from Cosmo Software (`http://cosmosoftware.com/`). Using this plug-in, which is available for Silicon Graphics, Windows and Macintosh platforms, you can »walk« or »fly« through the model and view the scene from all sides. If you look at the 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* (models): one shows the hits, files, cached files, sites and the amount of data sent by day and the other one shows the server's access 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 pre-defined *Viewpoints* (camera positions) and CosmoPlayer's *Navigation panel*. For customization use the CosmoPlayer pop-up menu.



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 on Friday between 16:00 and 17:00.



The 3D model (second scene)

INTERPRETATION OF THE RESULTS

http-analyze creates a summary of the information found in the server's logfile. The analyzer counts the requests, saves the unique URLs, sitenames, browser types and referrer-URLs and creates a comprehensive statistics report. The following terms are used in this report:

Hits (color: green) A hit is any response from the web server on behalf of a request sent by a browser, such as text (HTML) files, images, applets, audio/movie clips and even error messages. For example, if a page is requested which contains two inline images, the server would generate three hits: one hit for the HTML page itself and two hits for the images. If an invalid URL is requested, the server would respond with a Code 404 (*Not Found*) status code, which is also a response accounted for as a hit.

Files (color: blue) If the server sends back a file for this request, this is accounted for as a Code 200 (*OK*) response. Such a response is classified as a *file sent*. Again, file here means any kind of a file, no matter whether it contains text (HTML documents) or binary data (images, applets, movies, etc.). Note that if you would configure the web server to only log accesses to HTML files, but not images nor any other binary data, the number of files would directly correspond to the number of documents served.

Cached (color: yellow) A *cached file* is a Code 304 (*Not Modified*) response. This response is generated by the server if a document hasn't changed since the last time it was transferred to the site requesting it. If the browser has access to a local copy of a document requested by a user – either through its local disk cache or through a caching server –, it sends out a *conditional request*, which asks for the document to be sent only if it has been changed since it was requested the last time. If the document hasn't been change since then, the server sends back a *Code 304* response to inform the browser that it can use its local copy.

While this caching mechanism can significantly reduce network traffic, it causes an inaccuracy in the statistics report regarding the number a file is requested by someone because of two reasons: First, the browser can be configured to send conditional requests *every time*, *once per session* or *never* if a cached file is requested. Second, online services, ISPs, companies and many other organizations use so-called caching servers or proxies, which itself fulfill requests if the file is found in the cache. Since proxies can serve hundreds to thousands of users, requests from certain sites could be caused by thousands of users requesting a cached file or by just one person with his/her browser configured to not cache anything at all.

The ratio between *files sent* and *cached files* therefore reflects the efficiency of caching mechanisms – but only for those requests which were handled by your web server.

Pageviews
(color: magenta) The *pageview* mechanism can be used to separate requests for text or HTML files from all other types of requests. If a filename pattern has been defined, **http-analyze** classifies all URLs matching this pattern as pageviews (text files), which allows to estimate the number of »real« text documents transmitted by your web server. Filename patterns may be defined using the option **–G** or the **PageView** directive in the configuration file. The suffix **.html** is pre-defined already.

KBytes transferred
(color: orange) This is the amount of data sent during the whole summary period as reported by the server. Note that some servers record 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) log the size of the file instead the amount of data transmitted actually.

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 *cached files (Code 304)* responses with the size of the corresponding file. Because **http-analyze** can determine the size of a file only if the file has been transmitted at least once in the same summary period, the values for *KBytes saved by cache* and *KBytes requested* are just approximations of the real values.

Unique URLs

The total number of *unique URLs* is the sum of all different URLs (files) on your web server, which have been requested at least once in the corresponding summary period.

Referrer URLs

If a user follows a link to your web site and his/her browser sends the URL of the page containing the link to the server, this URL is logged as the *referrer URL* (the location referring to your document). Note that the browser does not necessarily send a referrer URL and even if it does, a proxy server may alter or delete it before forwarding the request to a web server. Such requests appear under *Unknown* in the referrer URL list.

Self-referrer URLs

As soon as the browser detects any inline objects (images, applets, etc.) in a page just loaded, it sends out separate requests for those objects. If the objects reside on the same server as the page referring to them, the corresponding referrer URLs contain the URL of the page on your server. Such requests are called *self-referrer URLs*. If configured correctly, **http-analyze** separates all self-referrer URLs from the rest of the referrer URLs in the report. This allows to separate accesses, which actually originated by using inline objects in a text page, from the remaining (external) accesses.

Unique sites

This is the number of all different hostnames or IP addresses found in the logfile. Each different hostname is counted only once per period, so this number shows how many systems did send requests to your server.

Sessions

(color: red) Similar to unique sites, this is the number of different hostnames or IP addresses accessing the server during a certain *time-window*, which defaults to one day for backward compatibility. Accesses from a known hostname outside this time-window get accounted for as a new *session*. You can increase or decrease the time-window for sessions using the option **–u** or the **Session** directive in the configuration file. For example, if you set the time-window to 2 hours, all accesses from the same host in less than 2 hours are accounted for as the same session, while any access more than 2 hours apart from the first one is accounted for 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 more information about a document such as its size without have the server send its 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 current HTML specification at `http://www.w3.org/` for information about all valid response codes and their meaning. **http-analyze** recognizes HTTP/1.1 responses according to RFC 2616.

Unresolved

A system identifies itself to a web server using an *IP number*. Depending on the configuration, the web server might perform a DNS lookup to resolve the IP number into a hostname. If no hostname has been assigned to this IP number, only the IP number is logged. Such requests are accounted for under *Unresolved* in the country list of the statistics report. Since some systems intentionally have no hostname, a percentage of up to 35% for unresolved IP numbers is absolutely normal.

If the country list shows only 100% unresolved IP numbers, either enable the DNS lookup in your web server or have a DNS resolver utility preprocess the logfile before feeding the data into **http-analyze**. For our Commercial Service Licensees, we offer a fast DNS resolver utility with negative caching and a history mechanism. Visit the support site at `http://support.netstore.de/` for more information.

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,
- find out the email address of a visitor,
- track the path a visitor takes through your site,
- measure the time a visitor sees a page of your server,
- determine the last page someone saw before leaving your site,
- inform you about the sudden death of the visitor while looking at your homepage,
- nor show any other information not recorded 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 real visitors of your server.

However, if you use an appropriate server structure with files grouped by its content or if you use the **HideURL** directive to group unstructered files together, the statistics report does show you at least 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 current directory or in the output directory specified at invocation of **http-analyze**. 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.

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 also 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 separate 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 output files of a full statistics report in a **www***YYYY* directory:

index.html
: is the main page for the year and contains the total numbers of *hits*, *files sent*, *cached 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 contains the values for the whole year, while the values for the last 12 months then will be continued in the index file for the new year. 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 numbers of *hits*, *files sent*, *cached 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 period. This list 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 that the country list is meaningful only for hostnames with 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, not **.com** ...

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 Silicon Graphics, Windows and Macintosh systems. See `http://cosmosoftware.com/` for more information. All 3D models are displayed in the *3D window*, so that you can compare them with the graphs in the conventional report.

topurl*MMYY***.html**, **topdom***MMYY***.html**, **topuag***MMYY***.html**, **topref***MMYY***.html**
: These 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 user agents, which have been used by visitors to access your web site. The user agent information is an identification sent by the browser and logged by the web server. Although the format for this identification is well-defined, it isn't obeyed by any browser. If possible, **http-analyze** reduces the name of the user agent in the *Top lists* to the browser type including the first digit of its version number. If it is not possible to isolate the browser type from the user agent, the full identification string as sent by the browser is stored.

  A referrer URL is the URL of the page containing a link to your web site, which has been followed by someone to reach your site. Note that for manually entered URLs no referrer URL gets logged. Also, some browsers do not send a referrer URL or send a faked one. Entries without a referrer URL are collected under *Unknown* in the referrer list. 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.png
: This small icon showing the graph from the main page is displayed on the main page under the base directory for each statistics report.

OPTIONS

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

–d *(daily mode)* generate a short statistics report for the current day 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 no history 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 are used only to create a summary page for the last 12 months. The timestamps from the logfile entries feed into **http-analyze** always take preceedence over any records in the history unless the option **–e** is specified.

–B create buttons only and exit. The analyzer copies or links the required files and buttons from the central directory **HA_LIBDIR** into the output directory specified by **–o**.

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

–X print the URL to file a bug report. Use command substitution or cut & paste to pass this URL to your favourite browser, complete the form and submit it.

–3 create a VRML 2.0-compliant 3D 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 resulting model.

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

–e use the history file even in full statistics (**–m**) mode. If this option is specified and you analyze the logfiles for several months in one run, **http-analyze** uses the results recorded in the history file for previous months and skips all logfile entries up to the first day of a new month not recorded in the history (usually the current month). This option is useful if you rotate your logfile once per quarter and want **http-analyze** to skip all entries for previous months which have been processed already.

–f create an additional frames-based user interface for the statistics report. This interface requires JavaScript.

–g *(generic interface)* create a conventional (non-frames) user interface for the statistics report without the optional JavaScript-based navigation window. By default, the conventional interface includes JavaScript enhancements for window control, which only become active if the user has enabled JavaScript in his/her browser. Use this option only to completely disable JavaScript enhancements in the report even if the user has enabled JavaScript in the browser.

–n *(no update)* do not update the history file. Since the history is used in the statistics report to create the main summary page with the results of last 12 months, this option must be used to not mess up the statistics report when analyzing logfiles for previous months (before the last one).

–q do not strip arguments to CGI scripts. By default, **http-analyze** strips arguments from CGI URLs to be able to lump them together. If your server creates dynamic HTML files through a CGI script, they are reduced to the URL of the script. If **–q** is specified, those argument lists are left intact and CGI URLs with different arguments are treated as different URLs. Note that this only works for requests to scripts, which receive their arguments using the **GET**, but not the **POST** method. See the section *Interpretation of the results* for an explanation of the request methods and the **StripCGI** directive.

–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, a dot is printed for each new day in the logfile.

–x list each image URL literally rather than lumping them together under the item *All images*. Without this option, **http-analyze** collects all requests for images *(*.gif, *.png, *.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.

–M MS IIS-Mode: use case-insensitive matching for URLs. This violates the standard, but is necessary for logfiles produced by IIS servers to correctly identify unique URLs.

–b *bufsize*

defines the size of the I/O buffer for reading the logfile (default: 64KB). Usually, the best size for I/O buffers is reported on a per-file base by the operating system, but some OS report the logical blocksize instead. If **http-analyze –v** reports a »Best buffer size for I/O« less than or equal to 8 KB, you should specify a size of 16 KB for pipes and up to 64 KB for disk files to increase the processing speed.

–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.

–i *newcfg*

create a new configuration file named *newcfg*. If an old configuration file was also specified using the **–c** option, older settings are retained in the new file. Any command line options take preceedence over old configuration file entries and will be transformed into the corresponding directive if possible. For example, specifying the output directory using the option **–o** *outdir* will produce an entry **OutputDir** *outdir* in the new configuration file.

–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`). This location can also be specified using the environment variable **HA_LIBDIR**.

–o *outdir*

use *outdir* instead of the current directory as the output directory for the statistics report. **http-analyze** checks automatically for the required files and buttons in *outdir*. If they are missing or out of date, the analyzer copies them from **HA_LIBDIR** into the output directory. See also the **OutputDir** and the **BtnSymlink** directives.

–p *prvdir*

defines the name of a »private« directory for the detailed lists of *files*, *sites*, *browsers* and *referrer URLs*. Because *prvdir* must reside directly under the output directory, its name may not contain any slashes ('/'). A private directory for detailed lists may be useful to restrict access to those lists if the rest of the statistics report is publicly available. Note that for restricting access to the complete statistics report, you do **not** need to place the detailed lists in a private directory. See also the **PrivateDir** directive.

–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 (cached files 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* defines 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 commas as in `-t 20U,0L,20S`. See also the **Top*** directives in the configuration file.

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

–w *hits* sets 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.

–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/99 -E Feb/99` restricts the analysis to January 1999.

–F *logfmt*
: the logfile format to use. Valid keywords for *logfmt* are **auto** for auto-sensing the logfile format, **clf** for the *Common Logfile Format*, or **dlf** and **elf** for the two variants of the *W3C Extended Logfile Format*. See also the section *Logfile Formats* above.

–L *lang* use the language *lang* for warning messages and for the statistics report. See also the directive **Language** and the section *Multi-National Language Support* for more information about localization of **http-analyze**.

–C *chrset*
: force use of *chrset* for the browser's encoding when displaying the statistics report. This is needed for languages which require special character sets such as Chinese. See also **HTMLCharSet** and the section about *Multi-National Language Support*.

–G *pattern,...*
: defines 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,...*
: defines 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**. To specify more than one name with a single **–H** option, use commas to separate them. See also the **IndexFiles** directive.

–O *vname,...*
: defines 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 *vname* doesn't include a protocol spcifier, two URLs with the `http` and the `https` protocol specifier are added for each name. To specify more than one server name with a single **–O** option, use commas to separate them. 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 may be suitable only for viewing on a fast system such as a 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 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.

–S *srvname*
: use *srvname* for the server name. If no server name is defined, **http-analyze** uses the hostname of the system it is running on. 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. Since **http-analyze** uses its built-in defaults if no TLD file is specified, you rarely will need this option. See also the **TLDFile** directive and the sample TLD file included in the distribution.

–U *srvurl*
: defines *srvurl* as the URL of the server to be used for hotlinks in URL lists. Useful if the report for your web server is published on another server. Also necessary for virtual servers to have **http-analyze** generate correct hypertext links in the report. See also the **ServerURL** directive.

–W *3Dwin*
: defines 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).

–Z *showdom*
: defines *showdom* as the number of components in a domain name which make up the organizational part. This is usually the *second-level domain*, so that the last two components of the domain name (for example, `company.com`) are used as the organizational part. However, some countries prefer to use *third-level domains*, so that the hostnames use 4 or more components, where the last 3 are used for the organizational part (as in `company.co.uk`). To recognize such third-level domains, *showdom* can be set to the value 3. Hostnames with exactly 3 components will still be reduced to their second-level domain if *showdom* is set to 3.

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.

Typical Usage

This is an example for the typical use of **http-analyze** on Unix systems:

```
$ http-analyze -v3f -o /usr/web/htdocs/stats /usr/ns-home/logs/access.log
```

On Windows systems, open a DOS window, change into the directory where you did install **http-analyze** and run a command similar to the following:

```
C:> http-analyze -v3f -o c:\web\htdocs\stats c:\programs\msiis\access.log
```

Note that on Windows systems, **http-analyze** searches for the required buttons and files in the subdirectory `files` of the current directory it is running in. Therefore, if you get error messages about missing buttons make sure you did change into the directory the analyzer is installed in (by default the installation directory is `C:\Programs\RENT-A-GURU\http-analyze2.4`).

CONFIGURATION FILE

You can define server-specific configuration settings for **http-analyze** in an *analyzer configuration file*. To have the analyzer use such a configuration file, specify its name with the option **–c** *cfgfile* or the environment variable **HA_CONFIG**. Note that command line options always take preceedence over settings in a configuration file.

If the option **–i** *newcfg* is specified, **http-analyze** creates a configuration template in the file *newcfg*. Any other command line options present will be transformed into its appropriate definitions in the new configuration file. The settings then can be customized further by manually editing the configuration definitions using a standard text editor.

To update an old configuration file into a new format, specify its name using the option **–c** in addition to **–i**. This will instruct the analyzer to retain any settings from the old 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 part of the string in an optional third field only. 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 (default: 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 either **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 associated with the top-level-domain. Example:

```
AddDomain           .compuserve.com     CompuServe
AddDomain           .aol.com            AOL
```

AuthURL *boolean value*
Defines whether accesses which required authentication should be skipped. By default, such URLs appear in the report just like ordinary URLs. If **AuthURL** is set to *Off*, *No*, *None*, *False*, or *0* the analyzer skips authenticated requests in the logfile, so that they will be suppressed from the statistics report. Example:

```
AuthURL             No
```

BtnSymlink *boolean value*
Creates symbolic links to the required buttons and files in **HA_LIBDIR** instead of copying them into the output directory. If you are going to analyze a large number of virtual servers which reside on the same host, you can probably save disk space by avoiding copies of all buttons and files into each output directory. Note that this directive can be used only on systems which support symbolic links. Example:

```
BtnSymlink          Yes
```

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

Defines 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 on white backgrounds (**CustLogoW**) and another one for use on black backgrounds (**CustLogoB**). The images should be approximately 72 × 72 pixels in size and must be placed into the buttons subdirectory of the central libdir (**HA_LIBDIR/btn**). Next time a report is generated, the analyzer copies those logos into the output directory and includes them in the report. Example:

```
CustLogoW  btn/mycompany_sw.png         http://www.mycompany.com/
CustLogoB  btn/mycompany_sb.png         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. Useful for virtual servers with a separate Document Root. **Note:** Do not define this directive to analyze the whole server. Explicitely setting **DocRoot** to '/' (the default) only increases processing time. Example:

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

HTMLCharSet *chrset*

Force use of *chrset* for the browser's encoding when displaying the statistics report (same as option **–C**). This is needed for languages which require special character sets such as Chinese. See also the section about *Multi-National Language Support*. Example:

```
HTMLCharSet         iso-8859-1
```

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

The HTML *prefix* and *trailer* to be inserted into the statistics output files at the top and bottom of the page. If defined, the **HTMLPrefix** string must include the <BODY> tag. To read the HTML code from a file, specify its name as the *prefix* or *trailer*. Example:

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

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,monospaced
```

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 for backward compatibility with older configuration files. Example:

```
HeadSize            4
SmallSize           2
```

HideAgent *agent string*

Hide a browser type under an arbitrary *string* (item). Needed only for a certain browser whose vendor still can't spell its 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 configuration file template produced by **http-analyze –i** for more examples hot to use the **HideRefer** directive. Example:

```
HideRefer           http://www.altavista.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 to be accounted for by appending it in square brackets to the *string* as shown in the last example below. Example:

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

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 as required by the HTTP standard. If the option **–M** is specified, URLs will become case-insensitive for compatibility with non-compliant web servers. Note that images are hidden automatically under *All images* by default unless **–x** is specified. Example:

```
HideURL             *.map               [All image maps]
HideURL             /robots.txt         [Robot control file]
HideURL             /newsletter/*       MyCompany Monthly Newsletter
HideURL             /products/*         MyCompany Products
HideURL             /~delta-t/          DELTA-t Homepage
HideURL             /~delta-t/*         DELTA-t more pages
```

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 ***.png**, **/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,*.png,*.jpg,*.jpeg
IgnURL          /stats/
```

IndexFiles *idxfile [,idxfile ...]*

Defines 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 *lang*

Use the language *lang* for warning messages and for the statistics report (same as option **–L**). See the section *Multi-National Language Support* for more information about localization of **http-analyze**. Example:

```
Language        de
```

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. If no logfile is specified, **http-analyze** always reads *stdin*. Example:

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

LogFormat *logfmt*

Use this logfile format. Valid values for *logfmt* are **auto** for auto-sensing the logfile format, **clf** for the *NCSA Common Logfile Format*, or **dlf** and **elf** for the two supported variants of the *W3C Extended Logfile Format*. See the section *Logfile Formats* for a detailed description of those formats. Example:

```
LogFormat       clf
```

MSIISmode *boolean value*

Use case-insensitive string comparison for URLs. Needed for MS IIS which makes no difference between upper- and lower-case characters. MS users may regard this as an enhancement, while for the rest of the world this is just a violation of the RFC 2616 HTTP standard and should be ignored. Example:

```
MSIISmode       Yes
```

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 using the default size of 420×190 pixels. Example:

```
NavWinSize      440x200
```

NavigFrame *size*

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

```
NavigFrame      140
```

NoiseLevel *hits*

Sets 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 of the statistics report should be created (same as option **–o**). By default, the output directory is the current directory. Example:

```
OutputDir           /usr/web/htdocs/stats
```

PageView *pattern [,pattern ...]*

Defines 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 *prvdir*

Defines the name of a »private« directory for the detailed lists of *files*, *sites*, *browsers* and *referrer URLs* (same as option **–p**). Because *prvdir* must reside directly under the output directory, its name may not contain any slashes ('/'). A private directory for detailed lists may be useful to restrict access to those lists if the rest of the statistics report is publicly available. Note that for restricting access to the complete statistics report, you do **not** need to place the detailed lists in a private directory. 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 hostname of the system it is running on. 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. Also necessary for 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
```

ShowDomain *number*

Defines the number of components in a domain name which make up the organizational part (same as option **–Z**). This is usually the *second-level domain*, so that the last two components of the domain name (for example, `company.com`) are used as the organizationial part. However, some countries prefer to use *third-level domains*, so that the hostnames use 4 or more components, where the last 3 are used for the organizational part (as in `company.co.uk`). To recognize such third-level domains, *ShowDomain* can be set to the value 3. Hostnames with exactly 3 components will still be reduced to their second-level domain if *ShowDomain* is set to 3. Example:

```
ShowDomain          3
```

StripCGI *boolean value*

Do not strip arguments to CGI scripts (same as option **–q**). By default, **http-analyze** strips arguments from CGI URLs to be able to lump them together. If your server creates dynamic HTML files through a CGI script, they are reduced to the URL of the script. If **StripCGI** is set to *Off*, *No*, *None*, *False* or *0*, those argument lists are left intact and CGI URLs with different arguments are treated as different URLs. Note that this only works for requests to scripts, which receive their arguments using the **GET**, but not the **POST** method. See the section *Interpretation of the results* for an explanation of the 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 (cached files 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. Since **http-analyze** uses its built-in defaults if no TLD file is specified, you rarely will need this directive. Example:

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

TblFormat *tblname specifier*

Defines the layout of tables in the statistics report. The argument *tblname* may be one of:

`Month`	for the statistics of the last 12 months (main page)
`Day`	for the daily statistics in the short and full summaries
`Load`	for the average load by weekday, hour, minute, second
`Country`	for the list of countries
`TopTen`	for all *Top N* lists
`Overview`	for all overviews
`Lists`	for all detailed lists (preformatted text)
`NotFound`	for the list of *NotFound* responses

The *specifier* string defines the items to be shown in the table:

`n`, `N`	an index number or label (don't touch!)
`h`, `H`	the number of *hits*
`f`, `F`	the number of *files sent*
`c`, `C`	the number of *cached files*
`p`, `P`	the number of *pageviews*
`s`, `S`	the number of *sessions*
`k`, `K`	the amount of *data sent* in Kbytes (integer value)
`B`	the amount of *data sent* in bytes (float value)
`L`	a dynamically created label (don't touch!)

If a format specifier is used in upper-case, the value displayed in the report will include the percentage for this number. Example:

```
TblFormat           Month     n h f c p s k
TblFormat           Day       n H F C P S k
TblFormat           Country   N H F P S k L
```

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 *vname,...*

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 *vname* doesn't include a protocol specifier, two URLs with the `http` and the `https` protocol specifier will be 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. See the section *Output files* for further information about this yearly model. Example:

```
VRMLProlog          3Dprolog.wrl
```

MULTI-NATIONAL LANGUAGE SUPPORT

http-analyze supports *Multi-National-Language-Support (MNLS)* according to the *X/Open Portability Guide (XPG4)* and the *System V Interface Definition (SVR4)*. For systems without MNLS support, a simple native implementation is used. See the file `INSTALL` included in the distribution for information about installation of the appropriate MNLS support for your system. The option **–V** displays the type of MNLS support compiled into a binary.

All text strings and messages of **http-analyze** are contained in a separate message catalog, which is read at start-up of the program. If a message catalog is installed in the system, you can select the language to be used for warning messages and for the statistics report by setting the appropriate *locale*. This can be done by defining the `LANG` (*XPG4/SVR4 MNLS*) or the `HA_LANG` (*native MNLS*) environment variable or by using the option **–L**. When using **–L**, the analyzer switches to the specified language when it has recognized the option. If no message catalog exists for the specified locale, **http-analyze** uses built-in messages in english language.

Certain languages require a specific character set to be used by the browser when displaying the statistics report. This can be defined using the option **–c** or the **CharSet** directive. The following table summarizes the most common combinations of languages and character sets. Note that the name of the locale is system-specific (for example, **de** could be **de-iso8859** on some systems.

Country	*Locale*	*Encoding*
Standard C	`C`	`us-ascii`
Arabic Countries	`ar`	`iso-8859-6`
Belarus	`be`	`iso-8859-5`
Bulgaria	`bg`	`iso-8859-5`
Czech Republic	`cs`	`iso-8859-2`
Denmark	`da`	`iso-8859-1`
Germany	`de`	`iso-8859-1`
Greece	`el`	`iso-8859-7`
Spain	`es`	`iso-8859-1`
Mexico	`es_MX`	`iso-8859-1`
Finland	`fi`	`iso-8859-1`
France	`fr`	`iso-8859-1`
Switzerland	`fr_CH`	`iso-8859-1`
Croatia	`hr`	`iso-8859-2`
Hungary	`hu`	`iso-8859-2`
Iceland	`is`	`iso-8859-1`
Italy	`it`	`iso-8859-1`
Israel	`iw`	`iso-8859-8`
Japan	`ja`	`Shift_JIS` or `iso-2022-jp`
Korea	`ko`	`EUC-kr` or `iso-2022-kr`
Netherlands	`nl`	`iso-8859-1`
Belgium	`nl_BE`	`iso-8859-1`
Norway	`no`	`iso-8859-1`
Poland	`pl`	`iso-8859-2`
Portugal	`pt`	`iso-8859-1`
Russia	`ru`	`KOI8-R` or `iso-8859-5`
Sweden	`sv`	`iso-8859-1`
Chinese	`zh`	`big5`

Since the message catalogs are independent from the base software, more languages may become available without having to re-compile or re-install the software. Please visit the homepage of **http-analyze** for up-to-date information about the available languages. For more information about localization, see *environ(5)* and *setlocale(3)* in the online manual.

EXAMPLES

After successful compilation of **http-analyze** you can test-run the analyzer before installing it permanently. Just create a subdirectory for the output files and run **http-analyze** on either one of the sample logfiles included in the distribution (as shown below) or use your web server's logfile. For example, to create a full statistics including a frames-based interface and a 3D VRML model in the subdirectory **testd**, use the following commands:

```
$ cd http-analyze2.4
$ mkdir testd
$ http-analyze -vm3f -o testd files/logfmt.elf
http-analyze 2.4 (IP22; IRIX 6.2; XPG4 MNLS; PNG)
Copyright 1999 by RENT-A-GURU(TM)
Generating full statistics in output directory 'testd'
Reading data from 'files/logfmt.elf'
Best blocksize for I/O is set to 64 KB
Hmm, looks like Extended Logfile Format (ELF)
Start new period at 01/Jan/1999
Creating VRML model for January 1999
Creating full statistics for January 1999
... processing URLs
... processing hostnames
... processing user agents
... processing referrer URLs
Total entries read: 8, processed: 8
Clear almost all counters at 03/Jan/1999
Start new period at 01/Feb/1999
No more hits since 02/Feb/1999
Creating VRML model for February 1999
Creating full statistics for February 1999
... processing URLs
... processing hostnames
... processing user agents
... processing referrer URLs
... updating 'www1999/index.html': last report is for February 1999
Total entries read: 3, processed: 3
Statistics complete until 28/Feb/1999
$
```

To view the statistics report, start your browser and open the file **testd/index.html**.

For permanent installation of **http-analyze**, issue a `make install` to copy the required files into the appropriate directory. The executable is usually installed in `/usr/local/bin`, while the required buttons and files are placed under `/usr/local/lib/http-analyze` unless this has been changed by defining the **HA_LIBDIR** make macro during installation.

Note that you do not need to install files in a new statistics output directory anymore if they have been installed in **HA_LIBDIR**; this is now done automatically by **http-analyze** if it runs the first time on this output directory.

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 1999 in the directory **/usr/web/htdocs/stats**:

```
$ cd /usr/ns-home/logs
$ http-analyze -vm3f -o /usr/web/htdocs/stats log1999/access.01
```

The next command uncompresses the logfiles for a whole year and feeds the data via a pipe into the analyzer, which then creates a statistics report for this period. All options are passed to the analyzer through a customized configuration file specified with **–c**:

```
$ gzcat log1998/access.[01]?.gz | http-analyze -c /usr/httpd/analyze.conf -
```

The following command creates a configuration file template with the name **sample.conf**. Any additional options will be transformed into the appropriate directives in the new configuration file. In this example, the server's name specified with **–S** is transformed into a **ServerName** directive and the output directory specified with **–o** is transformed into a **OutputDir** directive. All other directives are set to their respective default value. To further customize any settings, use a standard text editor.

```
$ http-analyze -i sample.conf -S www.myserver.com -o /usr/web/htdocs/stats
```

To update an old configuration file into the new format while retaining any old settings, specify its name when creating the new file. Again, command line options may be used to alter certain settings; they take preceedence over definitions in the old configuration file. The following command reads the file **oldfile.conf** and transforms its content into a new file named **newfile.conf**:

```
$ http-analyze -c oldfile.conf -i newfile.conf
```

REGULAR INVOCATION VIA CRON

Although **http-analyze** can be run manually to process logfiles, it usually is executed automatically on a regular base. On Unix systems you use the *cron(1)* utility, while Windows systems provide a similar functionality with the *AT* command. To have your statistics report updated automatically, use the following scheme:

1) Install a cron job which calls **http-analyze –m3f** to create a full statistics report once per hour or twice per day depending on the processing load caused by analyzing the logfile. Note that the full statistics report is created for the first time at the second day of a new month.

2) Optionally install a cron job which calls **http-analyze –d** more often to create a short statistics report. Although this will only update the *Hits by day* section of the report, the advantage of the short statistics mode is that **http-analyze** needs only a fraction of the time required to create a full statistics report. However, this is only needed if the total time needed to create full statistics reports requires more than 15 minutes.

3) Install a shell script which rotates (saves) the server's logfile, restarts the web 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 how to do this for several virtual web servers at once.

4) Because of 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 »noise« appearing at the end of a logfile. However, to initialize the files for the new month, you should run **http-analyze –m3f** on the logfile for the current month immediately after the statistics for the previous month have been generated.

Note that all cron jobs must run with the user ID of the owner of the output directory except for **rotate-httpd**, which must run with the user ID of the server user. This is a sample *crontab*(1) for the scheme described above:

```
# Generate a full statistics report twice per day at 01:17 and 13:17
17  1,13 * * *  /usr/local/bin/http-analyze -m3f -c /usr/httpd/analyze.conf

# Generate a short statistics report 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 logfiles at the first day of a new month at 00:00
0 0 1 * *       /usr/local/bin/rotate-httpd
```

PERFORMANCE CONSIDERATIONS

The processing time needed to create full statistics reports depends on many factors:

- The size of the I/O buffer (reported by **http-analyze** when **–v** is given) should be as big as possible. For example, a buffer size of 64 KB can significantly reduce disk activity when reading the logfile.
- If many **Ign*** directives are defined, the analyzer must compare each logfile entry against each entry in the corresponding **Ign*** list. The recommended way to suppress certain parts of the web server in the statistics report is to have the server not record any accesses to those areas in the logfile. Similar, many **Hide*** directives may also require additional table lookups, although this will happen only once for each unique (different) URLs, sitename, browser type or referrer URL.
- If **StripCGI** is set to **No**, this will require more memory.
- Some systems impose a memory limit on a per-process base (see *ulimit(1)* and *setrlimit(3)*). There are no unusual requirements regarding main memory needed by **http-analyze** – to be precise that means »the bigger, the better« –, but you should make sure that about 5-10 MB is available for processing of a medium-size logfile.

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 files/logfmt.elf
http-analyze 2.4 (IP22; IRIX 6.2; XPG4 MNLS; PNG)
Copyright 1999 by RENT-A-GURU(TM)
Generating full statistics in output directory 'testd'
Reading data from 'files/logfmt.elf'
Best blocksize for I/O is set to 64 KB
Hmm, looks like Extended Logfile Format (ELF)
      1 01/Jan/1999:16:37:25 [298971279], req="GET /", sz=280 <- OK (Code 200), PAGEVIEW
Start new period at 01/Jan/1999
      2 01/Jan/1999:16:38:39 [298971355], req="GET /def/", sz=910 <- OK (Code 200), PAGEVIEW
      3 02/Jan/1999:16:39:39 [299060697], req="GET /abc/", sz=910 <- OK (Code 200), PAGEVIEW
...
```

Filing bug-reports

If you want to file a bug report, use the option **–X** to have **http-analyze** generate an URL of a bug reporting form with some information already filled in. You can pass this URL to your favourite browser using cut & paste or – on Unix systems – using command substitution as in:

```
$ netscape `http-analyze -X`
```

This address a bug report form on `http://support.netstore.de/` with the following information filled in already:

- the customer's name as specified in the registration
- the registration ID with licensing information (Personal/Commercial License)
- the version number of **http-analyze**
- the platform the program was compiled for.

Using this interface to submit report bugs will ensure proper handling and timely response. Please note that although we gladly accept bug reports from everyone, only Commercial Service Licensees are entitled to request technical assistance or open a support call.

REGISTRATION

http-analyze is available through our web site for evaluation purposes. In the evaluation version an »unregistered version« 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 the »unregistered version« button to follow the link to our online 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 version« button by email. In the free 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 its authorized resellers. You will receive a registration ID and two registration images as replacements for the »unregistered version« button 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 note, which appears only once on the main page and inside the navigation frame. On all other pages, your company's name is shown. Additionally, you can add your company's logo to the report using the **CustLogoW** and **CustLogoB** directives in the configuration file, which are enabled in the commercial version only. Except for this feature and the individual support for Commercial Service Licensees, both versions of the software 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, the registration information applies to all virtual servers being analyzed.

To brand the software, detach the registration images we sent to you from the email. After detaching them, there should be two files `free-netstore_s[bw].png` for the free version and `comm-netstore_s[bw].png` for the commercial version. Next, define the **HA_LIBDIR** environment variable if you did choose another directory for the central libdir rather than the default (`/usr/local/lib/http-analyze`). For example, if you can't become *root*, you would choose a directory for which you have write permissions, install the analyzer files there and then use the **HA_LIBDIR** variable to pass its name to **http-analyze**. Finally, brand the software by executing the following command as root:

```
# http-analyze -r "Customer Name" regID type
Registration information saved in file '/usr/local/lib/http-analyze/REGID'
#
```

where *Customer Name* is the name of the organization this license is registered for, *regID* is the registration ID of the license and *type* is either the keyword `free` or `comm` according to the type of the license. Now run the analyzer to have the new buttons appear in the statistics report.

Note that running the analyzer the first time will install or update any older buttons and files in the statistics output directory automatically; there is no need to run some helper application as it was the case in previous versions of **http-analyze**.

YEAR 2000 COMPLIANCE

All versions 2.X and above of **http-analyze** are fully Year 2000 compliant. There will be no problems with date-related functions after the year 1999 as long as the operating system itself is Year 2000 compliant also. 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. **http-analyze** is provided under the terms of the license agreement included in each distribution.

Please see `http://www.netstore.de/Supply/http-analyze/year2000.html` for more information about the Year 2000 compliance real-time tests we did run with **http-analyze**.

DATE USAGE IN HTTP-ANALYZE

The analyzer depends on the timestamp found in the logfile entries produced by a web server. For the *NCSA Common Logfile Format and the W3C Extended Logfile Format* a Year 2000 compliant date format was choosen from the 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 Operating System (see below).

To retain compatibility with previous versions of the log analyzer, **http-analyze** generates two-digit years in some output filenames. However, those files are placed in a subdirectory containing the year in four digits, which makes all output filenames fully Year 2000 compliant.

The date format in the **–I** and **–E** options allows specification of a year using only two digits. **http-analyze** interprets values greater and equal to 69 in 1900 and values lower than 69 in 2000. This way, the analyzer covers the whole range of the time representation in modern Operating Systems. However, any year can always be specified unambiguously by using four digits.

DATE USAGE IN THE OPERATING SYSTEM

Rumors has it that some systems don't recognize the Year 2000 as a leap year. Although **http-analyze** computes leap years for itself correctly, it maps dates into weekdays using the *localtime(3)* function, which might fail if the OS doesn't recognize the Year 2000 as a leap year.

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. 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 versions.

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)
LANG	language to use if XPG4 MNLS support is compiled in (see **–V**)
HA_LANG	language to use if native MNLS support is compiled in (see **–V**)

FILES

The following required files are installed in the library directory as defined by the environment variable **HA_LIBDIR** or the hard-coded default defined at compile-time. See also the section *Statistics Report* above for the names of the HTML output files.

btn/.png*	buttons files used in the statistics report
TLD	list of all top-level-domains
ha2.0_*.png	**http-analyze** logos for your web site (for black and white bg)
logfmt.[cde]lf	sample logfiles in CLF, DLF and ELF format
*3D**	required files for VRML model

SEE ALSO

rotate-httpd	shell script to rotate the web server's logfiles
http://www.netstore.de/Supply/http-analyze/	homepage of **http-analyze**
http://support.netstore.de/	support site of **http-analyze**

NOTES

Logfile entries must be sorted in chronological order (ascending date) when feed into the analyzer. 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. To sort the data from several different logfiles into a chronologically sorted data stream, we provide a utility `ha-sort` to our Commercial Service Licensees.

To increase response time of web servers, DNS lookups are often disabled. In this case **http-analyze** does not see any hostname, but only numerical IP addresses. To resolve the IP addresses into hostnames, we provide a very fast DNS resolver `ipresolve` to our Commercial Service Licensees, which does negative caching and saves all data in a history file.

Please visit our support site at `http://support.netstore.de/` for more information about the available helper applications.

COPYRIGHT

CREDITS

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