Volume 3

Hyperwave Administrator's Guide

Hyperwave Information Server

Version 4.1

HYPERWAVE
Organize your future

Hyperwave Administrator's Guide

Hyperwave
Information Management Inc.
2350 Mission College Blvd.
Santa Clara, CA 95054
USA
Tel: 1-408-982-8228
Fax: 1-408-982-8229
Email: info@hyperwave.com
http://www.hyperwave.com

Hyperwave
Information Management GmbH
Stefan-George-Ring 19
D-81929 Munich
Germany
Tel.: ++ 49 89 993074-0
Fax: ++ 49 89 993074-99
Email: info@hyperwave.de
http://www.hyperwave.de

Document Version 1.0

TABLE OF CONTENTS

INTRODUCTION

The *Hyperwave Administrator's Guide* contains information about every aspect of server administration from configuring the server to uploading documents. It is intended for people who want to set up and administrate a Hyperwave Information Server. For a general description of Hyperwave's features and instructions on how to use Web browsers to upload documents to Hyperwave, see the *Hyperwave User's Guide*. If you are interested in programming to expand the capabilities of your server, refer to the *Hyperwave Programmer's Guide*. See the *Hyperwave Installation Guide* for instructions on installing the server.

You will find a brief summary of the contents of each chapter of the *Administrator's Guide* below.

Chapter 1 - Server Organization

The Hyperwave Information Server architecture is briefly introduced here and the different modules of the server and their tasks are described. The location of the modules and their corresponding log files in the directory structure is also described.

Chapter 2 - Configuring the Server

The GUI configuration tool WaveSetup is presented here. Hyperwave's configuration file `.db.contr.rc` is also described in detail.

Chapter 3 - Maintenance and Administration

This chapter introduces you to those tools which have to do with maintenance and administration of a Hyperwave Information Server, e.g. tools for backing up and restoring a server or creating new user accounts. Two special forms of user identification, user mapping (automatic identification depending on the host the client accesses the server from) and external identification (using an existing user database instead of creating accounts for each user), are described. Further, this chapter describes the Hyperwave Server Pool, WaveBack, using NetDynamics with the server, etc.

Chapter 4 - Document Management

Those command line tools that have to do with document management (uploading and downloading, copying, deleting, using version control, etc.) are described in this chapter.

Chapter 5 - Troubleshooting

This is the chapter to refer to if you have difficulties with the server. Common problems and their solutions are presented. This chapter also includes references to further information about Hyperwave and tells you who to contact for support.

1 SERVER ORGANIZATION

This chapter gives an overview of the architecture of the Hyperwave Information Server. The function of each server module, as well as how the modules interact with each other, is explained. The location of the different parts of the server in the directory structure is also described.

1.1 SERVER ARCHITECTURE

Though it may look like it to the user, the Hyperwave Information Server is not just one big block of software. It is in fact a group of modules, each of which has specific tasks to fulfill and which are able to communicate among each other. This modular design makes Hyperwave flexible: modules may be added on to the server or existing modules may be replaced as desired. The individual processes are started, monitored and restarted if necessary by a control process called `hwservercontrol`, which can be configured to start any process along with the server (see Configuring the Server with .db.contr.rc, page 14).

The server can be seen as being made up of three different layers: the *protocol conversion layer*, the *session layer*, and the *database layer*. The protocol conversion layer makes Hyperwave a multi-protocol server. The WWW gateway (known as WaveMaster) transforms HTTP to Hyperwave's client-server protocol (HG-CSP), so that lower layers need to deal only with a single protocol. The session layer communicates with the database layer (sometimes indirectly via a gateway). An instance of the session layer is created for every client connection, to retain state information and to parallelize client requests. The database layer is where the actual documents, links and meta-information are stored. This layer consists of three modules: the **object server** (`wavestore` or Oracle, depending on your Hyperwave installation), which creates, modifies and deletes objects (links and collections) and their relationships, indexes them for searching, and manages users and access permissions, the **full text server** (`ftserver`), which maintains an inverted index of all text documents for searching, and the **document cache server** (`dcserver`), which stores the local documents of the server, as well as cached documents from remote servers.

The following section gives a more detailed description of the various parts of the server.

1.1.1 WWW GATEWAY (WAVEMASTER)

The WaveMaster is a very important part of the Hyperwave Information Server because it resembles the system's "front end," that is, it defines the visual appearance for the user. Its task is to enable users with web clients (like Netscape or Internet Explorer) to access information in Hyperwave.

This is not as simple and straightforward as it sounds. The gateway has not only to transform the protocols (from connectionless and stateless HTTP to connection-oriented and stateful HG-CSP) and map Hyperwave's separated links to HTML's embedded links, but also to define the presentation details of collections, clusters, search forms and buttons for navigation.

Figure 1 shows what a Hyperwave collection looks like through the WaveMaster. The collection itself, and all user interface and navigation elements are assembled into a single HTML document, which is then visualized by the web client. The toolbar at the top of the page contains menus with

items for various functions such as user identification, setting options (for example, preferred language), searching, help, returning to the home collection, inserting documents, etc.

A key feature of WaveMaster is its flexibility. The appearance of the user interface is determined by a special programming language called PLACE. Hyperwave administrators can alter the PLACE templates that come with the server to suit their individual needs (see the *Hyperwave Programmer's Guide*). Certain aspects (such as user interface language, quality of documents received) can even be configured by the client, that is, the user.

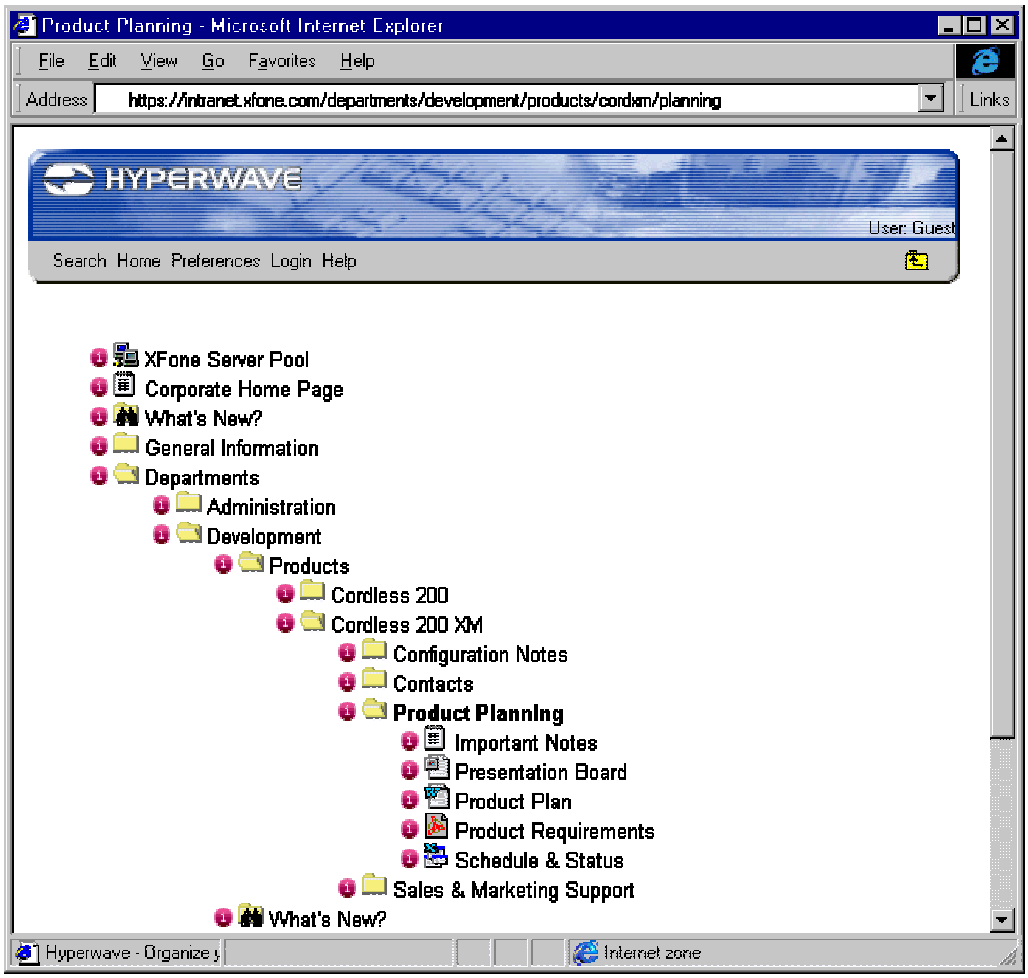


Figure 1: Viewing a Hyperwave collection with WaveMaster

WaveMaster is useful not only for browsing information, it is also possible to use it to upload and edit information, or perform a search on a Hyperwave Information Server. This means that you can author information in Hyperwave using any web client (see the *Hyperwave User's Handbook* for details).

1.1.2 SESSION LAYER (HGSERVER)

The session layer communicates with the client using the Hyperwave Client-Server Protocol (HG-CSP).

The connection-oriented HG-CSP protocol naturally supports an efficient server architecture, where one server process is started per client connection only at the beginning of a session. In this

case, this process is `hgserver`, a lightweight process (a process that uses almost no memory) that is created by a master copy listening on port 418 (the well-known port number reserved for Hyperwave), whenever a client connects. The process is terminated when the client connection is closed, or when no client requests have been received for a certain timeout interval.

At the beginning of the session, `hgserver` negotiates communication parameters with the client, and remembers them for the remaining session, that is, it keeps some state information, including user identification data.

The `hgserver` process has no direct access to persistent data. Rather, it is itself a client of the servers in the database layer, most importantly the object server (`wavestore` or Oracle). Because the low-level databases process transactions serially it is important that the `hgserver` splits operations that might take a long time into a number of small (fast) transactions for the database servers. In this way, requests from one client do not block execution of requests from other clients, as they are processed in parallel. `hgserver` also employs a small cache of often-used objects (for example, the current user record) to further speed things up.

When accessing objects from remote Hyperwave Information Servers, `hgserver` opens connections to the other servers (using HG-CSP, that is, looking like a client to the other server), forwards the requests to the other servers, assembles the results and passes them on to the client. The number of parallel connections to other servers is limited to 10 (if more are needed, the connection least recently used is closed to open the new one).

As an example of a fairly complex request, let us consider the deletion of a text document from a certain collection (using `wavestore` as object database). The client specifies the ids of the document and the collection.

1. First, `hgserver` needs to check access permissions by looking at the user record and possibly the **Rights** attribute of the collection object (see page 63). In other words, it has to retrieve the collection object from `wavestore`. Let us assume access permissions are granted.
2. A quick look at the document-collection relation (again something for `wavestore`) tells `hgserver` whether the document is also a member of other collections or resides on a remote server (which implies that there is another parent collection on the remote server). If so, `wavestore` is instructed just to remove the document from the collection as requested (by removing the corresponding entry in the document-collection relation), and the client request is finished. If the document was on a remote server, this server is informed so that it can update its document-collection relation, too.
3. If, however, the collection is the only parent of the document, the document has to be physically removed.
   a. In order to do this, the document object is retrieved from `wavestore`. The **DocumentType** attribute reveals that it is a text document. The **Path** attribute contains a handle for the actual document in `dcserver`.
   b. Because it is a text document, `ftserver` is instructed to remove the words occurring in the document from its indexes. Since these indexes are organized to find the OIDs a given word occurs in, and not the words of a given OID, `hgserver` has to retrieve the document from `dcserver` and pass it on to `ftserver`.
   c. Next, `dcserver` is told to physically remove the document, freeing up the space it used to occupy.
   d. Finally, `wavestore` is instructed to remove the document object. In an atomic transaction, it also removes the entry in the document-collection relation as well as all the anchor objects (source and destination) attached to the document. Anchors pointing to the document are flagged as "open"

Although the whole operation is fairly complex and may take some time (up to a few seconds, depending on the size of the document), the individual requests in particular for `wavestore` are

rather small, so they do not block parallel operations of other clients for an undue amount of time (a few milliseconds). In the case of a server crash in the middle of the operation (for example, as the result of a power failure), we could theoretically end up with the document removed from `ftserver` or `dcserver`, but not from `wavestore`. However, the special transaction logging performed by `hgserver` makes sure that the whole operation can be completed as part of the recovery process which takes place when the server is restarted, so that the server databases are always in a consistent state.

1.1.3 DATABASE LAYER

1.1.3.1 OBJECT SERVER (WAVESTORE OR ORACLE)

The native Hyperwave object server (`wavestore`) is a multi-user, read-write database. It stores a database of objects (in the file `Object.db`) and relations of objects (in the `*.rel` files). The other files in the server directory are indexes derived from these files for faster access and searching, and log files. Write operations always append to the database files. Deleted records are not removed physically until the next database reorganization.

The objects in the database include documents, collections, anchors, user records, user groups and server descriptions. The relations define collection membership and links. In fact, the link information of all documents is contained completely in the object server.

ORACLE DATABASE

Starting with version 4.0 of Hyperwave, it is possible to use an Oracle database instead of Hyperwave's native database. This new feature can provide several advantages, the most important of which is that you can use an existing Oracle database under Hyperwave. A further advantage is that you can distribute your database over several hard disks when using Oracle, something which is not available when using Hyperwave with its native database.

1.1.3.2 FULL TEXT SERVER (FTSERVER)

The full text server stores indexes of word lists. Whenever a new text document is inserted, a reference to it sent to `ftserver` by `hgserver`, the `ftserver` gets the document by means of that reference, and its words are inserted into the index data structure. Likewise, when the document is deleted, its words are removed from the index. Unlike in many other systems, indexing is performed immediately. Subsequently querying `ftserver` with a list of words ("terms") results in a ranked list of objectIDs and scores.

The full text server currently consists of two engines (the native engine and Verity), of which you must choose one. See the chapter on configuring Hyperwave Information Server to find out how to do this.

The native engine employs one index per language. Before going into the index, the text is passed through a set of filters. One is a language-specific, configurable list of stop words which contains words that occur in texts very frequently and which should not be indexed. The other is a stemmer - also language dependent - which cuts off common suffixes (so that, for example, "computer" also matches "computers"). Also, words are converted to lower case before indexing.

Verity is a commercial product which has been transparently incorporated into `ftserver`. See page 19 for information on configuring Verity for your server.

1.1.3.3 DOCUMENT CACHE SERVER (DCSERVER)

The document cache server is where the actual documents on a Hyperwave Information Server are kept. It has several tasks:

- to serve local documents to the client
- to accept documents from the client and store them

- to fetch documents from remote servers, pass them on to the client and store them in a local cache
- to serve previously cached documents to the client
- to run CGI scripts and send their output to the client.

Like `wavestore`, there is only one `dcserver` process but it keeps a connection open to each `hgserver` process. Usually, however, data is sent to the client over a separate connection. There are two ways of doing this:

- with the "old" method, where the client opens an arbitrary port and sends the number to `dcserver` over the `hgserver` connection. `dcserver` then opens a connection to this port. The problem with this method is that it doesn't work if the client is behind a firewall.
- The "new" method solves the problem with the firewall. With this method, the client opens the connection to a known port (4712) and fetches the data through it. In this case the firewall just has to be configured so that it allows connections to port 4712.

1.2 DIRECTORY STRUCTURE

As described above, the server consists of several parts, including `wavemaster`, `hgserver`, `wavestore` (or Oracle, depending on your installation), `dcserver` and `ftserver`. When Hyperwave is installed, the installation program creates a directory in `hwsystem`'s home directory for each of these modules and gives it the corresponding name.

The server is configured using the WaveSetup tool (see page 8) or by directly editing the `.db.contr.rc` file, which is found in `hwsystem`'s home directory (see page 14).

LOG FILES

The control process for the server, `hwservercontrol`, produces a log file (`~hwsystem/log/server.log` by default). When there is a problem, this file can usually tell you in which module the problem occurred, and is the log file you should look at first when there is a problem. You should continue searching for the cause of the problem by looking into the log file that corresponds to the appropriate module. `hg.log`, `dc.log`, `wavestore.log`, `ft.log`, and `wave.log` are the default log file names for the `hgserver`, `dcserver`, `wavestore`, `ftserver` and `wavemaster` log files respectively. These files are found in the directory `$HOME/log`.

DCSERVER DIRECTORY

There are three directories in the `dcserver` directory:

- `local`, where the documents of the local server reside
- `cache`, where the cached documents reside
- `cgi`, which, by default, contains a directory `cgi-bin`, where you may put your CGI scripts.

FTSERVER DIRECTORY

The `ftserver` directory contains several files and directories:

- `rep0001.raw` / `rep0001.map` / `rep0001.dirty`

  These make up the internal documents table which contains references to the documents being indexed.

- `configuration`

  This directory contains stoplists for English, German and Japanese where the configuration files for the native full text engine are located.

- `inv`

  The directory where the inverted indexes of the native fulltext engine are located.

- `verity`

  The directory where the Verity configuration files and the Verity indexes are located. See page 18 for more information on Verity.

2 CONFIGURING THE SERVER

This chapter introduces the easy-to-use GUI tool WaveSetup, which can be used to configure the Hyperwave Information Server. All variables contained in Hyperwave's main configuration file .db.contr.rc are also described.

2.1 CONFIGURING THE SERVER WITH WAVESETUP

WaveSetup is an easy-to-use graphical tool which divides the configurable aspects of the server into six categories: **General**, **Network**, **Searching**, **Appearance**, **Performance**, and **Logging**. All settings are explained in detail in WaveSetup's interface (see Figure 2) which makes the tool simple to use.

The following instructions explain how to access WaveSetup for the Windows NT and UNIX platforms:

UNIX

1. Connect to your server with a Web browser on port 9999, e.g. `http://intranet.myserver.com:9999`.
2. The WaveSetup tool appears and you can use it to configure your server. When you are finished, you must stop and start the server before the changes take effect.

The user name and password for WaveSetup depend on certain conditions:

3. In most cases, the user name and password are taken from the UNIX account under which Hyperwave Information Server was installed.
4. In the case of UNIX systems with shadow passwords, the password field in `/etc/passwd` cannot be read, and thus the password of the UNIX account cannot be used. In this case, the user name is taken from the account and the password is "hwsystem". For security reasons it is recommended to change the password as soon as possible using WaveSetup.

Note: The user name and password for the server and WaveSetup are completely independent, i.e. changing one does not change the other.

WINDOWS NT

1. Connect to your server with a Web browser on port 9999, e.g. `http://intranet.myserver.com:9999`.
2. The WaveSetup tool appears and you can use it to configure your server. When you are finished, you must stop and start the server before the changes take effect.

The original user name/password combination for WaveSetup in the NT version is `hwsystem`/`hwsystem`. The password should be changed as soon as possible using the WaveSetup tool.

Note: *You must stop and start the server again for the changes you made with WaveSetup to take effect. This can be done by clicking on the* Restart server now *button on the* General *page of WaveSetup.*

Below is a detailed descriptions of the aspects of the server that can be configured using WaveSetup, with references to further information in this guide.

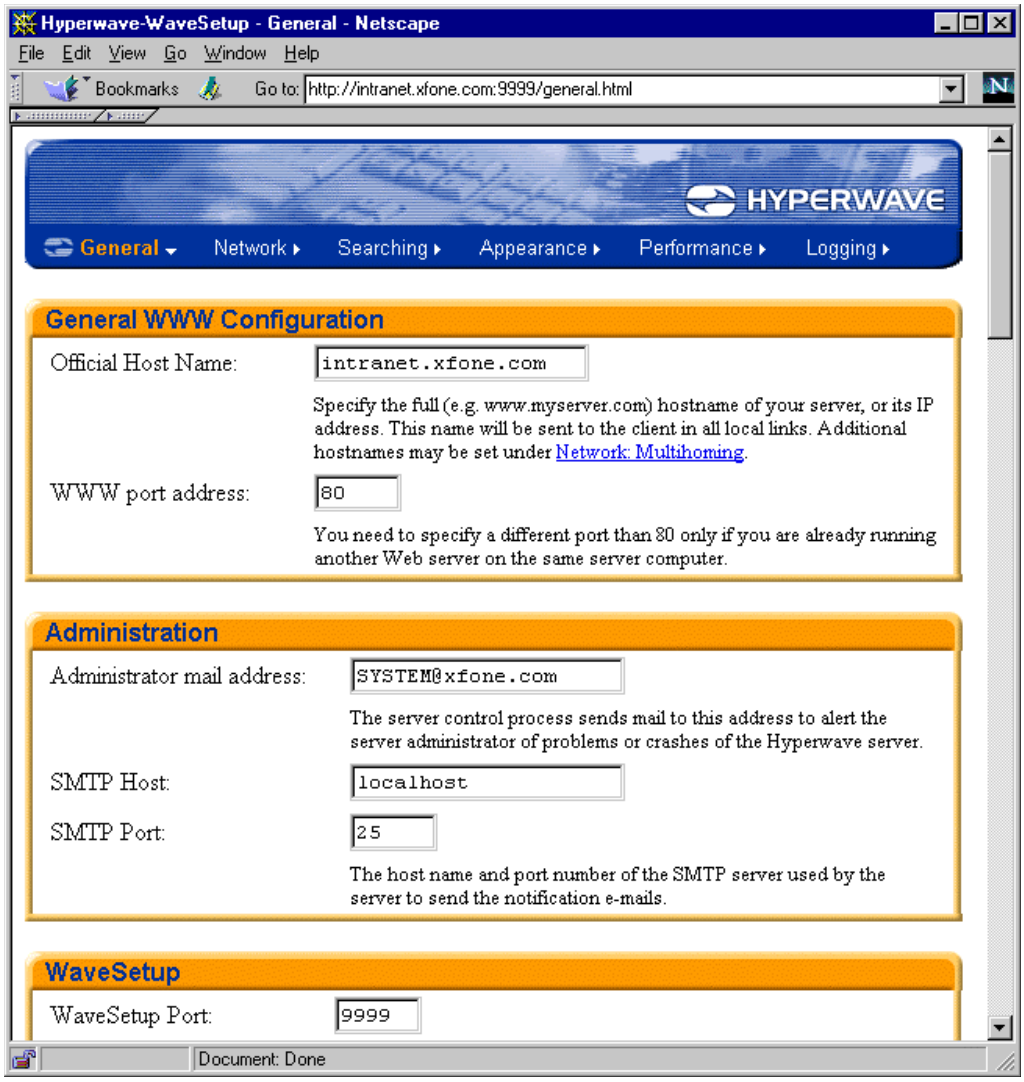


Figure 2: The WaveSetup tool

2.1.1 GENERAL

GENERAL WWW CONFIGURATION

- **Official Host Name**: Specify the full (e.g. www.myserver.com) host name of your server, or its IP address. This name will be sent to the client in all local links. Additional host names may be set under Network: Multihoming.
- **WWW port address**: You need to specify a port other than 80 only if you are already running another Web server on the same server computer.

ADMINISTRATION

- **Administrator mail address**: The server control process sends mail to this address to alert the server administrator of problems or crashes of the server.
- **SMTP host** and **SMTP port**: The host name and port number of the SMTP server used by the server to send the notification e-mails.

WAVESETUP

- **WaveSetup Port**: Choose any port that is not being used by another program. Under UNIX, you cannot use ports less than 1024 (unless you run WaveSetup as root user).

- **WaveSetup User**: The user allowed to connect to WaveSetup.
- **WaveSetup Password**: The password needed to connect to WaveSetup.
- **Confirm Password**: Retype the password here.
- **Allowed Domains**: Domain names (separated by commas) from which connections to WaveSetup are allowed. Partial domain names (e.g. .hyperwave.com) can be used. Connections from the local host are always allowed and need not be configured.
- **Allowed IP addresses**: IP addresses (separated by commas) from which connections to WaveSetup are allowed. Connections from the local host are always allowed and need not be configured.

EXTERNAL AUTHENTICATION

- **Use external authentication**: If this is enabled the server also allows identification of users from the operating system's user database.

SERVER-SERVER COMMUNICATION

- **Server-Server Communication**: Hyperwave Information Servers can communicate with each other to maintain link consistency. There are three modes of operation available:
  - **None**: This is a stand-alone server with no communication to other servers.
  - **Server Pool**: A server pool is a group whose members communicate with each other. Server pools need additional configuration done by the hwpoolmanager tool.

See page 76 for more on how to configure server pools.

SERVER BACKUP

This section of WaveSetup configures the backup procedure for the Windows NT Hyperwave Information Server.

- **Automatic Backup**: Switches on/off automatic server backup.
- **Start time**: The time when to start the automatic backup.
- **Interval**: How often to start the automatic backup. Choose a predefined value from the list.
- **Backup mail address**: The backup process sends a notification mail to this address.
- **Backup directory**: The directory in which the backup is stored/from where the backup is restored.
- **Backup command**: The actual tool doing the backup.
- **Backup command line**: The whole command line to do the actual backup. Normally, you should not have to change this. For a detailed explanation see the Administrator's Guide.
- **Restore command**: The actual tool doing the restore.
- **Restore command line**: The whole command line to do the actual restore. Normally, you should not have to change this. For a detailed explanation see the Administrator's Guide.

SET SERVER LICENSE

Type in or paste your new server license here (include everything from the line `----BEGIN HYPERWAVE LICENSE-----` to the line `----END HYPERWAVE LICENSE-----`). The license will be checked and saved at the correct location.

2.1.2 NETWORK

PERSISTENT CONNECTIONS

- **Keep alive**: Switch this off if you experience problems with older clients.
- **HTTP timeout**: Time (in seconds) an idle HTTP connection is kept open.

- **Cookie file name**: Cookies are the preferred method for maintaining session information. The cookie file lists clients known to be able or known to be unable (preceded by a "!") to handle cookies. Clients which do not support cookies get URLs with session keys (unless they are robots). You may directly edit the contents of the file.

ROBOT SETTINGS

- **Robot user**: The user name for robot requests.
- **Robot entry point**: The entry point (collection name) to the collection hierarchy for robot requests.
- **Robots file name**: The robots file contains a list of known robots. A client is searched in the robots file only if it is not found in the clients file. Robots get a special Hyperwave session with a special user and entry point. This database is maintained automatically.
- **Clients file name**: The clients file contains a list of known standard clients. Clients found in the client file are not searched for in the robots file. You may directly edit the contents of the file.

USER MAPPING

- **User mapping file**: The server can be configured so that users who connect to the server using a specific machine or a machine in a specific domain are automatically identified under a specified user name. The lines of the mapping file have the form "user: host host...;". The host may be either an IP address or a full or partial domain name (for example *.hyperwave.com). You may directly edit the file's contents with WaveSetup. See page 65 for more details on user mapping.

DEFAULT SESSION POOL

Hyperwave creates pools for all users who are either anonymous or who logged in through user mapping or multihoming. The pools limit the number of sessions that can be started by these users.

Additional session pools with different characteristics may be configured under Network: Session pools.

- **Minimum**: Minimum number of sessions in the pool.
- **Maximum**: Maximum number of sessions in the pool.
- **Start**: Number of sessions at startup.
- **Timeout**: Timeout (in seconds) when idle sessions are closed.

SESSION POOLS

You may configure session pools for specific user names. They are created automatically at server startup.

Warning: Do not configure session pools for users who are not logged in through user mapping or multihoming.

To add a pool, enter the user, the minimum sessions, maximum sessions, startup sessions and timeout for the pool.

MULTIHOMING

Hyperwave is able to assign HTTP requests to different users and entry points, according to the host entry or IP address they are connecting to. This can be used to run multiple virtual hosts on a single Hyperwave Information Server. A "*" as Hostname or IP address means "any".

To use multihoming, enter a **Hostname**, **IP address**, **User** and **Entry point**.

See also the `.db.contr.rc` variable `<scope>::HOSTS`.

2.1.3 SEARCHING

FULLTEXT INDEXING

Fulltext engine: Choose the fulltext engine to be used for fulltext indexing and searching.

The following engines are currently supported:

- Native: The native Hyperwave fulltext indexer.
- Verity: The Verity Search'97 fulltext engine. Verity is available for Windows NT, Digital UNIX (OSF/1), IBM AIX, Irix, HP-UX 10.x, Sun Solaris and Sun OS.

Note: Changing the fulltext engine causes the fulltext server to rebuild its indexes, which may take some time.

Enable: If you are inserting or removing a large number of documents you may want to switch off immediate fulltext indexing for performance reasons. The fulltext index is updated when fulltext indexing is switched on again. Note that updating the fulltext index may take some time.

Verity Locales: Verity needs a configuration directory containing language specific settings for each supported language; it is relative to $HOME/verity/common. Additionally, for each language the character set to be used can be configured.

SUBSCRIPTIONS

- **Subscriptions**: Enable subscriptions.
- **Sender**: The sender appearing in the "From:" field of the notification mails.
- **Interval**: The time interval Wavenotify searches for query objects in minutes.

Note: Subscriptions are enabled by default when a new server is installed. However, if your server is an update of an older server, subscriptions must be turned on manually.

FULLTEXT FILTERS

External filters can be used to index arbitrary document types with the native fulltext engine (they have no effect if you're using Verity). The document type must be specified as MIME type and language, separated by a comma (eg. application/msword,en).

To add a filter, enter the **MIME type** and the **Command line**.

CUSTOM INDEXES

Indexes for arbitrary attributes may be built so these attributes can be searched. There are four types of indexes:

- Keyword: value is one or more words separated by spaces, hyphens or periods which are indexed separately.
- LangKeyword: similiar to keyword but the words have a language associated with them.
- NameKey: value is a character string which is not separated.
- Time: value is a time ([YY]YY/MM/DD).

To add an index, enter the **Field name** and the **Index type**.

See page 44 for more information about configuring custom indexes.

2.1.4 APPEARANCE

MULTILINGUALITY

- **Default language**: The default language for user interactions.

PLACE TEMPLATES

- **Template and icon path**: The directory where PLACE template files and icon images reside.
- **Master template file**: The master PLACE template file to be used.
- **User PLACE templates**:
  - **Groups**: Insert one or more group names separated by spaces. When configured, only users in these groups are allowed to use user PLACE templates.
  - **Enable**: When switched on all users are allowed to use user PLACE templates. This switch overrides the group setting.

Note: If you make changes in your PLACE templates, you can use the "Reread Templates" *button at the bottom of this page to cause Wavemaster to reread the templates.*

2.1.5 PERFORMANCE

TRANSACTION SYNC

- **Disable**: If this is disabled, the object server synchronizes its database files with the physical disk only every 30 seconds, otherwise after every transaction. Enabling transaction sync slows down insertion of objects, but guarantees integrity of the database even after a power loss in the middle of a modification. Default is off.

DOMAIN NAME SERVICE

- **Number of DNS resolvers**: This many DNS lookups can be performed in parallel. DNS lookups are performed on each client request.

CACHE SETTINGS

The cache settings control how much RAM is used to speed up disk accesses and how much disk space is used to keep local copies of remote documents. You probably won't need to modify the default values, unless you run a very large server (in which case you should increase them) or a very small server (in which case you can decrease them). The defaults are good for servers containing about 100,000 documents.

- **Object Server**: Specified in blocks of 4KB. Used to cache search indexes in RAM for faster access to objects. Default is 1,000 blocks (a bit less than 4 MB).
- **Fulltext Server**: Specified in blocks of 4KB. Used to cache search indexes in RAM for faster fulltext queries. Default is 1,000 blocks (a bit less than 4 MB).
- **Document Server**: Specified in Bytes. The dcserver uses this many bytes to store local copies of documents from other Hyperwave or HTTP servers. Default is 1,000,000 Bytes. Should be increased if you run a server pool.
- **Wavemaster Cache**
  - **Enable**: If on, Wavemaster caches documents, which makes subsequent accesses to cached documents faster.
  - **Cache size**: The total size (in kilobytes) of the cache.
  - **Maximum element size**: Specified in kilobytes. Documents larger than this value are not cached.
  - **Cache timeout**: Specified in seconds. Documents are removed from the cache after that time.

DOCUMENT SERVER TIMEOUTS

- **TCP read**: Time in seconds a TCP connection to be read from can be idle before it is closed. Default is 120 (2 minutes).
- **TCP write**: Time in seconds a TCP connection to be written to can block before transmission is aborted. This is NOT a maximum for the total transmission time. Default is 120 (2 minutes).
- **CGI**: Time in seconds a CGI script may be idle, i.e. does not generate any output, before it is terminated. Default is 120 (2 minutes).
- **Control connection**: Time in seconds after which an idle control connection (to hgserver) is closed. Default is 0 (don't close).

2.1.6 LOGGING

LOGGING

The various server components produce log files for statistics and debugging purposes. If logging is turned on, there are three options how to treat old logfiles when a server component restarts:

- append to the existing file
- overwrite the existing file
- rename the existing file (by adding a time stamp) and start a new one

For each of the following Hyperwave processes, you can enter the name and full path of the log file and select one of the options described above for treating old log files:

- Wavemaster (native format)
- Wavemaster (common log format)
- Wavemaster (combined log format)
- Control program
- Object Server
- Document Server
- Fulltext Server
- HgServer
- Wavenotify
- HWBackup

2.1.7 DATABASE

This page appears in WaveSetup only if you are using Hyperwave Information Server with Oracle. The following parameters can be configured:

- The user name of the Oracle user (required)
- The password of the Oracle user (required)
- **DB String**: This is required if you are not connecting to the default data base (name of appropriate TNS entry).
- The PCTFREE, PCTUSED, INITIAL, NEXT and PCTINCREASE parameters for the HWOBJECTSOUP table.

2.2 CONFIGURING THE SERVER WITH .DB.CONTR.RC

It is also possible to configure the server by directly editing Hyperwave's main configuration file `.db.contr.rc`. This section lists every configurable variable in `.db.contr.rc`. The file is found in Hyperwave's home directory (usually `~hwsystem`). The variables are categorized according the server module which they affect. For more information about the different server

modules and their tasks, see page 2. The variable names are case-insensitive while their values are case-sensitive.

2.2.1 CONFIGURING HWSERVERCONTROL

`hwservercontrol` starts and controls all server processes. Its scope is `MAIN`.

`MAIN::LOG` : This variable tells `hwservercontrol` where to put its log file and tells it what to do with old log files when the server is restarted. The value is the name and path of the log file and is `$HOME/log/server.log` by default. Possible parameters are `no_rename`, which produces one big log file, and `remove_old`, which causes the old log file to be removed each time the `MAIN` process is started. If there is no parameter, then each time the server is started the old log file is renamed by adding a time stamp, so the old files have names like `server.log.33577e1b`. You may delete these old log files if you like.

Example: `MAIN::LOG[no_rename]= $HOME/log/server.log`

`MAIN::MAILTO`: Email address that is used to alert the server administrator of problems or crashes of the server. By default this is the name of the user who started the server plus the server name, e.g. `hwsystem@www.yourserver.com`.

`MAIN::PROCESS`: This is used to make `hwservercontrol` start arbitrary processes along with the server. The value of this variable is whatever you want to name your process. This declared name is then used as scope in the following variables:

`*::COMMAND`: The command line that starts the process.

`*::WDIR`: The current working directory of the process.

`*::PORT`: The port the process is listening to.

`*::LOG`: The log file of the process. See `MAIN::LOG` above for the parameters that can be used with this variable.

`MAIN::SMTPHOST` and `MAIN::SMTPPORT`: Sometimes `hwservercontrol` has to send notifications by mail. These variables configure the host and port for the SMTP server (the server used to send emails). Under UNIX this usually does not have to be configured (the default "localhost:25" should work).

`MAIN::TIMEDPROCESS`: This is used to make `hwservercontrol` start arbitrary timed processes along with the server. The value of this variable is whatever you want to name your process. This declared name is then used as scope in the following variables:

`*::COMMAND`: The command line that starts the process.

`*::WDIR`: The current working directory of the process.

`*::PORT`: The port the process is listening to.

`*::LOG`: The log file of the process. See `MAIN::LOG` above for the parameters that can be used with this variable.

`*::START[`*interval*`]`: The value of this variable is a start time in the format `YYYY/MM/DD hh:mm:ss`, and tells when the process should be started for the first time. If the date part of the start time is not given, the current date is used. If the start time is not given at all, the current date and current time are used. The *interval* is either "once", "hourly", "daily", "weekly" or "monthly", "mo", "tu", etc. (to start the process every Monday, Tuesday, etc.) or a number (for the number of minutes).

Example: `MAIN::START[th]=1998/01/03`

`MAIN::TIMEOUT`: This variable is used to configure the amount of time between TERM and KILL when `hwservercontrol` is stopping the server. Its value is a time in seconds. When the server is being stopped, `hwservercontrol` first tries to end all processes with the TERM signal (or its Windows equivalent); if the process does not react and terminate within a certain time interval, a KILL signal (or its Windows equivalent) is sent to it. The process cannot ignore the KILL signal. It is also possible to give `hwservercontrol` the timeout parameter when starting it with `hwstart`, e.g. `hwstart –timeout 5`.

`MAIN::WAVESETUP`: This Boolean variable tells `hwservercontrol` whether or not it should start WaveSetup along with the server. Its value can be "true" or "false".

2.2.2 CONFIGURING WAVESTORE/WAVEORACLE

`wavestore` is the native object database of the Hyperwave Information Server (see page 5). Starting with version 4.0 of Hyperwave, it is possible to use an Oracle database in place of the native database.

The `.db.contr.rc` variables that are used to configure `wavestore` have the scope `WAVESTORE`, while those for the Oracle database have the scope `WAVEORACLE`.

WAVESTORE VARIABLES

The following variable can only be applied to `wavestore`:

`WAVESTORE::COMPLEXTRANSACTIONS`: When this Boolean variable is turned on, the database is synchronized only every 30 seconds, which results in better insertion performance. Default is `TRUE`.

WAVESTORE/WAVEORACLE VARIABLES

The following variables can be applied to either `wavestore` or Oracle, where <*scope*> is either `WAVESTORE` or `WAVEORACLE` respectively.

<*scope*>`::CUSTOMINDEX`: A new custom index is created for each instance of this variable listed in the configuration file. See page 44 for a full description of custom attributes and how to use this variable.

<*scope*>`::LOG`: This variable tells `wavestore` or Oracle where to put its log file and tells `hwservercontrol` what to do with old log files when the server is restarted. The value is the name and path of the log file. The default value is `$HOME/log/db.log`. Possible parameters are `no_rename`, which produces one big log file, and `remove_old`, which causes the old log file to be removed each time the MAIN process is started. If there is no parameter, then each time `ftserver` is started the old log file is renamed by adding a time stamp, so the old files have names like `db.log.33577e1b`. You may delete these old log files if you like.

Example:

```
WAVESTORE::LOG[remove_old]=$HOME/wavestore.log
```

<*scope*>`::OBJECT_CACHE_SIZE`: The size of the object cache measured in number of objects. The minimum value is 2000.

Default: <*scope*>`:: OBJECT_CACHE_SIZE = 2000`

<*scope*>`::STOPLISTS`: This Boolean variable is used to turn on or off the use of stoplists during keyqueries.

2.2.3 CONFIGURING DCSERVER

`dcserver` is the document cache server (see page 5). The following variables are recognized:

`DCSERVER::CACHE[size]`: The maximum size of the document cache in bytes. The size given here counts only the sum of the individual documents, management information is not included. However, management information is negligible under normal circumstances.

Default: `DCSERVER::CACHE[size] = 10000000`.

`DCSERVER::LOG`: This variable tells `dcserver` where to put its log file and tells `hwservercontrol` what to do with old log files when the server is restarted. The value is the name and path of the log file. The default value is `$HOME/log/dc.log`. Possible parameters are `no_rename`, which produces one big log file, and `remove_old`, which causes the old log file to be removed each time the MAIN process is started. If there is no parameter, then each time `dcserver` is started the old log file is renamed by adding a time stamp, so the old files have names like `dc.log.33577e1b`. You may delete these old log files if you like.

Example:

```
DCSERVER::LOG[remove_old]=$HOME/dcserver.log
```

`DCSERVER::TIMEOUT[from]`: The timeout in seconds a TCP connection to be read from can be idle. If nothing arrives within this time it is considered to be an error.

Default: `DCSERVER::TIMEOUT[from] = 120`.

`DCSERVER::TIMEOUT[to]`: The timeout in seconds a TCP connection to be written to can block. If it blocks longer it is considered to be an error.

Default: `DCSERVER::TIMEOUT[to] = 120`.

`DCSERVER::TIMEOUT[cgi]`: The timeout in seconds a CGI script may be idle. Idle means that `dcserver` does not receive any output from a CGI script during that period. In this case the script is considered hung and an error is reported.

Default: `DCSERVER::TIMEOUT[cgi] = 120`.

`DCSERVER::TIMEOUT[rpcctl]`: The timeout in seconds after which an idle control connection is closed. Default is no timeout.

`DCSERVER::HTTP[User-agent]`: The value of the user agent field in the HTTP header of a request when acting as HTTP client.

Default: `DCSERVER::HTTP[User-agent] = "Hyperwave Document Server (dcserver)"`.

`DCSERVER::CGI[SERVER_SOFTWARE]`: The string to be set in the corresponding CGI environment variable, if not already set. Usually setting this variable will have no effect because WaveMaster sets it before when issuing `dcserver` to execute a CGI script.

Default: `DCSERVER::CGI[SERVER_SOFTWARE] = "Hyperwave Document Server (dcserver)"`.

`DCSERVER::CGI[timeout]`: The same as `DCSERVER::TIMEOUT[cgi]`. If these two variables have different values, the value of the one that appears later in the file is taken.

`DCSERVER::LIMIT[core]`: If your system supports setrlimit(), this value (in bytes) sets the maximum coredump size for debugging purposes.

Default: `DCSERVER::LIMIT[core] = 0`.

`DCSERVER::LIMIT[data]`: If your system supports setrlimit(), this value (in bytes) sets the maximum heapsize of `dcserver`. The default and also maximum in this case depend on your system.

2.2.4 CONFIGURING FTSERVER

`ftserver` is the full text server (see page 5). Configuration of the `ftserver` is logically divided into two separated classes: the overall configuration, and the engine specific. The overall configuration has the form

`FTSERVER::XXX =` *<some value>*

where XXX is some configuration item. The engine specific configuration usually has the form

`FTSERVER::`*<engine>*`_XXX` = *<some value>*

where *<engine>* is the module identifier. Currently supported engines are "native" and "verity". The native engine is supported on all architectures. For information on what architectures Verity is supported, see below.

2.2.4.1 VERITY

It is possible to use Verity instead of Hyperwave's native `ftserver` to index documents. Using Verity has several advantages, the most important of which is that it can filter many document types that `ftserver` cannot (e.g. MS Word, MS Excel, PDF, PostScript, etc.), without having to configure external filters. Verity is also faster because the document filters are built in and no separate processes have to be started when a special type of document has to be filtered.

When Hyperwave is installed for the first time, Verity is also installed and switched on. If you update the server, Verity is installed but the fulltext engine you were using before remains switched on. If you would like to switch Verity on or off, you can do so using WaveSetup or the server's main configuration file `.db.contr.rc` (see below).

Note: Verity is not available for the Linux and BSDI platforms.

SWITCHING VERITY ON WITH WAVESETUP

To switch the Verity search engine on with WaveSetup, do the following:

1. Access WaveSetup with a browser with the URL `http://`*<your_server_name>*`:9999`
2. Select WaveSetup's **Searching** tab.
3. Select "Verity" in the listbox next to "Fulltext Engine".
4. Click on the **Update Searching Settings** button to make the changes in Hyperwave's configuration file.
5. **Stop and restart your server** by clicking on the **Restart server now** button. This must be done for the changes to take effect.

SWITCHING VERITY ON WITH .DB.CONTR.RC

In `.db.contr.rc` enter the line

```
FTSERVER::ENGINE = verity
```

to turn on Verity. The possible values for `FTSERVER::ENGINE` are "verity" for Verity and "native" for the default search engine. **`ftserver` must be stopped and restarted before the change takes effect.**

VERITY'S UNIVERSAL FILTER

Verity uses a so-called *universal filter*, which recognizes several document types by default. This filter consists of

- the zone filter, which filters HTML documents
- the PDF filter, for Acrobat PDF documents (this filter is not available for the Dec Alpha platform)
- the KeyView Filter Kit V1.0, which supports filtering of WYSIWYG documents. The suites and document formats listed below are supported.

Suites

Suite	Version(s)
Microsoft Office	95, 97

Word Processing Documents

Format	Version(s)
Applix Words	4.2
Corel WordPerfect for Windows	5.x, 6, 7, 8
Corel WordPerfect for Macintosh	2, 3
HTML	3.2 compliant
Lotus AMI Pro	2, 3
Lotus AMI Professional Write Plus	All versions
Lotus Word Pro	96, 97
Lotus 1-2-3 (DOS/Windows)	2.0, 3.0, 4.0, 5.0
Lotus 1-2-3 (OS/2)	Release 2
Microsoft Rich Text Format (RTF)	1.x, 2.0
Microsoft Word for Windows	2, 6, 95, 97
Microsoft Word for DOS	4, 5, 6
Microsoft Word for Mac	4.0, 5.0, 6.0
Microsoft Works	All versions
Microsoft Write	All versions
Text files	N/A
XYWrite	4.12

Spreadsheets

Format	Version(s)
Corel Quattro Pro	7, 8
Lotus 1-2-3	2, 3, 4, 5, 96, 97
Microsoft Excel	3, 4, 5, 95, 97
Microsoft Works Spreadsheet	All versions

Presentation Graphics

Format	Version(s)
Corel Presentations	7.0, 8.0
Lotus Freelance	96, 97
Microsoft PowerPoint	4.0, 95, 97

The universal filter is configurable. It has a configuration file that tells it how to filter each type of document that it sees. It also allows multiple filters on each document, so that you are not limited to a single type of filter. The goal of the universal filter design, however, was to be able to filter all important document types "out of the box." That is, the default configuration file that ships with the search engine should be sufficient for almost all documents that you might want to index. Configuration is offered in case you have special needs that are not addressed in the standard configuration file.

CONFIGURING VERITY IN .DB.CONTR.RC

The variables for configuring Verity in `.db.contr.rc` are described in section Engine specific ftserver configuration on page 20.

REBUILDING INDEXES AFTER SWITCHING FULLTEXT ENGINES

If you switch between the native and Verity engines, you should have the indexes rebuilt. This is done by creating an empty file called `REBUILD` in `$HOME/ftserver` before restarting the fulltext server. This only needs to be done if you made the switch by directly editing the variables in `.db.contr.rc`. If you are using WaveSetup to switch between engines, the rebuilding of the indexes is done automatically.

2.2.4.2 OVERALL FTSERVER CONFIGURATION

`FTSERVER::LOG`: This variable tells `ftserver` where to put its log file and tells `hwservercontrol` what to do with old log files when the server is restarted. Its value is the name and path of the log file. The default value is `$HOME/log/ft.log`. Possible parameters are `no_rename`, which produces one big log file, and `remove_old`, which causes the old log file to be removed each time the MAIN process is started. If there is no parameter, then each time `ftserver` is started the old log file is renamed by adding a time stamp, so the old files have names like `ft.log.33577e1b`. You can remove these old files if you like.

Example:

```
FTSERVER::LOG[remove_old]=$HOME/ftserver.log
```

`FTSERVER::ENGINE`: This variable selects the engine to be used. The possible values are "native" and "verity". You may specify more than one engine, but only the last line takes effect. The default value is "verity". If you have engine specific configuration items specified, of course only the ones in this class are used and all others are ignored.

Example:

```
FTSERVER::ENGINE = verity
```

`FTSERVER::DUMMY`: This variable causes `ftserver` to pretend to index and remove documents but not actually do so. This can be useful when uploading large numbers of documents since not indexing documents at the time of insertion saves time. Documents which were not indexed can be indexed afterwards. `DUMMY` is a Boolean variable and is turned off by default.

Examples:

```
FTSERVER::DUMMY = true
FTSERVER::DUMMY = false
FTSERVER::DUMMY = 1
FTSERVER::DUMMY = 0
```

2.2.4.3 ENGINE SPECIFIC FTSERVER CONFIGURATION

NATIVE ENGINE

`FTSERVER::NATIVE_INDEXCACHESIZE`: This variable configures the maximum number of index blocks to be held in memory. One block occupies 4K. The default is 1000 blocks.

Example:

```
FTSERVER::NATIVE_INDEXCACHESIZE = 4000
```

(e.g. for a faster index)

`FTSERVER::NATIVE_SPLITTER`: This is a parameterized variable which configures an external program as a document-into-word splitter. The value is a command line as interpreted by `/bin/sh` and should be a filter which reads the document from standard input and writes the words (separated by white spaces) to standard output. The parameter is made up of two parts which are separated by a comma. The first part is the MIME type of the document. The second part is the language of the document. The default splitters are HTML parsers for all languages, as well as a plain text parser.

Example:

```
FTSERVER::NATIVE_SPLITTER[application/msword,en] = strings
```

This line causes `ftserver` to pipe English MS Word documents into the `strings` command, read the output, split it into separate words, pass them into several filters such as a stoplist, and index them.

VERITY ENGINE

FTSERVER::VERITY_MIMEMAP: Verity uses certain MIME types which are different from the ones Hyperwave uses. Therefore, there is an internal table which maps Hyperwave's MIME types to those which Verity understands. Entries are in the form FTSERVER::VERITY_MIMEMAP[*hyperwave_mimetype*] = *verity mimetype*. Currently the table contains the following entries by default:

```
FTSERVER::VERITY_MIMEMAP[application/vnd.ms-word] = application/msword
FTSERVER::VERITY_MIMEMAP[application/vnd.ms-excel] = application/x-ms-excel
FTSERVER::VERITY_MIMEMAP[application/vnd.ms-powerpoint] = application/x-ms-
powerpoint
```

FTSERVER::VERITY_CHARSET[*language*]: This variable is used to configure the character set that Verity uses for a language. *language* can be a Hyperwave language prefix (see page 89) or an ISO 639 language tag. Verity supports "850" (for DOS Codepage 850) and "8859" (for ISO 8859-1).

FTSERVER::VERITY_FILELOCK: A Boolean value for switching on/off file locking. Default is "on". This is mainly for testing purposes (because when server parts are running on different machines, file locking via NFS may cause problems). It may also be switched off to increase performance.

Example:

```
FTSERVER::VERITY_FILELOCK = off
```

FTSERVER::VERITY_KNOWLEDGEBASE : This is used to configure a knowledge base. Its value is a file name.

FTSERVER::VERITY_LOCALE[*language*]: This variable is used to configure a directory (relative to $HOME/verity/common) which contains a language configuration for Verity for a particular language. *language* can be a Hyperwave language prefix (see page 89) or an ISO 639 language tag.

Example: `FTSERVER::VERITY_LOCALE[ge] = german`

FTSERVER::VERITY_MAINTENANCE: Verity does maintenance and optimizations in the background when there are no requests to be handled. With this flag this background activities can be switched on or off (default is on).

Example: `FTSERVER::VERITY_MAINTENANCE = off`

FTSERVER::VERITY_MAXFILES: This variable designates the number of files Verity can open.

FTSERVER::VERITY_MAXMEM: This variable is used to limit the temporary memory used by Verity. Its value is a number representing the desired number of kilobytes.

FTSERVER::VERITY_MULTITHREAD: A Boolean value for switching on/off multithreading (default is off).

Example: `FTSERVER::VERITY_MULTITHREAD = on`

FTSERVER::VERITY_REBUILD_SYNC: This Boolean variable causes Verity to index documents only during the rebuild phase when it has the value on.

FTSERVER::VERITY_REJECTMIME: The value of this variable is a MIME type. With this variable it is possible to tell Verity not to index a certain MIME type. Incomplete MIME types can also be used, e.g., `VERITY_REJECTMIME = video`. This variable can be used in conjunction with FTSERVER::VERITY_WANTMIME. For example, by configuring VERITY_WANTMIME=image/gif and VERITY_REJECTMIME=image, you can make it so that Hyperwave only indexes GIFs and no other type of image.

FTSERVER::VERITY_SYNC: This Boolean variable is used to switch on and off the immediate indexing of documents that are uploaded to the server.

FTSERVER::VERITY_TOPICS: This variable is used to configure a topic set. Its value is a file name.

FTSERVER::VERITY_WANTMIME: The value of this variable is a MIME type. With this variable it is possible to tell Verity to index a certain MIME type. Incomplete MIME types can also be used, e.g., VERITY_REJECTMIME = video. This variable can be used in conjunction with FTSERVER::VERITY_REJECTMIME. For example, by configuring VERITY_WANTMIME=image/gif and VERITY_REJECTMIME=image, you can make it so that Hyperwave only indexes GIFs and no other type of image. Note that, for a given MIME type, if there is neither a VERITY_REJECTMIME nor a VERITY_WANTMIME configured for it, documents of that MIME type are indexed.

2.2.5 CONFIGURING HGSERVER

The following variable is currently recognized by hgserver.

HGSERVER::LOG: This variable is used to tell hgserver where to put its log file and tell hwservercontrol what to do with old log files when the server is restarted. The value is the name and path of the log file. The default value is $HOME/log/hg.log. This variable uses the parameters remove_old and no_rename (see FTSERVER::LOG).

2.2.6 CONFIGURING WAVEMASTER

In this section you will find the basics of WaveMaster configuration as far as it concerns the behavior of the server.

Until now you have read about the database and communication layer processes of the server that have to be running and are therefore started implicitly. WaveMaster is part of the protocol conversion layer and therefore is not really needed for server operation. For this reason it is not started implicitly by the watchdog and needs a process definition in .db.contr.rc. Nevertheless the server comes with a predefined WaveMaster configuration.

WaveMaster need not be a single process, there can also be multiple WaveMasters running on the same machine. They either have to use different ports or in case of multihoming they can use different IP addresses or both. For this reason, one entry in .db.contr.rc exists for every WaveMaster process that is started.

WaveMaster is aware of a configuration file which uses scoped variable syntax. By default it looks for .db.contr.rc which can also be overridden. The following entries are recognized by WaveMaster (in the following list <*scope*> stands for the scope of the WaveMaster process, for example WAVEMASTER):

<*scope*>::ADD_ICON[*MIME_type*]: This is used to configure an icon for a particular MIME type. Its value is an icon, given by a path relative to the wavemaster directory of your server, i.e. the value is a physical file name and not an object on the server.

Example: WAVEMASTER::ADD_ICON[application/x-hmcard] = icons/hmcard.gif

<*scope*>::ADD_POOLENTRY: This variable is used to configure a session pool for a specific user. The pool is started when the server is started. The value of the variable is five values separated by spaces: <*username*> <*min number of sessions*> <*max number of sessions*> <*number of startup sessions*> <*timeout*>. See also page 47.

<*scope*>::BINARY_DEPENDENT: Sometimes WaveMaster has to check other binaries (hgserver, wavestore, etc.) when it has dependencies on them. This variable tells it whether or not it can do so and should be set to TRUE if the binaries are on the same machine and FALSE if they are on a different machine.

Default: <*scope*>::BINARY_DEPENDENT = TRUE

<*scope*>::CACHE_CONFIG: This variable configures the cache with respect to cache size, size of the largest element that can be put into the cache, and the amount of time an element remains in the cache. Its value is three integer values separated by spaces: <cachesize (in bytes)> <maxelementsize (in bytes)> <time to live (in seconds)>.

Default: <*scope*>::CACHE_CONFIG = 10000000 1000000 600

<*scope*>::CACHE_CONTROL: This variable turns caching on and off. Its possible values are "true" or "false".

Default: <*scope*>::CACHE_CONTROL = true

<*scope*>::CGI_PATH: This tells the WaveMaster the name of the directory where the CGI scripts are kept so that it knows where the CGI script name attribute in URLs ends. See also page 112.

Example: <*scope*>::CGI_PATH = cgi-bin/

<*scope*>::CLIENTS_FILE: This file contains information about which clients are considered standard clients rather than robots. The order of client/robot detection is to first look at the CLIENTS_FILE to see if it is a client, and if it cannot be found there, to look at the ROBOTS_FILE to see if it is a robot. If the client cannot be found in either file it is considered to be a standard client. The clients file is supplied with the server and should not be changed.

Default: <*scope*>::CLIENTS_FILE = $HOME/wavemaster/idx/clients.idx

<*scope*>::COOKIE_FILE: The location of the file that is used to detect clients that can handle cookies.

Default: <*scope*>::COOKIE_FILE = $HOME/wavemaster/idx/cookies.idx

<*scope*>::CORE_SIZE: The maximum size in bytes of the coredump file if WaveMaster crashes.

Default: <*scope*>::CORE_SIZE = 10000000

<*scope*>::DEFAULT_LANGUAGE: Hyperwave is a multilingual system and you can specify WaveMaster's default interface language using this variable. The value of this variable must be one of the language abbreviations found on page 89.

Default: <*scope*>::DEFAULT_LANGUAGE = en

<*scope*>::DEFAULT_POOLENTRY: The pool entry used to create a session pool for a user when no specific pool has been configured for it (see page 47).

Default: <*scope*>::DEFAULT_POOLENTRY = default 1 5 1 600

<*scope*>::DEFAULT_ROBOT: This variable takes two strings - the user name and the entry point in the collection hierarchy for robot requests. This enables you to let robot requests run under a different account, with a different standard entry point, and even with different access rights than standard user requests. The use of this setting even goes as far as to give robots indexing the server additional descriptions from additional collections especially prepared for them.

Default: <*scope*>::DEFAULT_ROBOT = anonymous /

<*scope*>::DNS_RESOLVERS: To achieve nonblocking DNS lookup, WaveMaster has a certain number of DNS resolving slaves. The number of slaves is set through this variable. The minimum value is one.

Default: <*scope*>::DNS_RESOLVERS = 3

<*scope*>::HELP_COLLECTION: The collection to get online help from. If this variable is not defined, help has to be provided in the PLACE template file.

Default: <*scope*>::HELP_COLLECTION = <*undefined*>

<*scope*>::HOSTNAME: The name of the host WaveMaster is running on. This host name is taken to create a URL from a URN on the fly. The entry overrides the OFFHOSTNAME environment variable.

Default: <*scope*>::HOSTNAME = <*name.of.local.host*>

<*scope*>::HOSTS: Specifies user name mappings and entry points used for multihoming. This variable can be multiply defined. A definition takes four entries separated by spaces: <*host name*> <*IP address*> <*mapped username*> <*entry point*>. Wildcards ('*') are valid for *host name* and *IP address* but not for parts of them! Host names must be valid host names for the computer your server is running on. Dealing with host names and IP addresses is a two-level mechanism: first WaveMaster looks at the HTTP request and determines whether it contains a host entry. If the request does contain a host entry, it matches the entry against the HOST variables until it finds the correct entry. If not, WaveMaster matches the IP address it is running on against the HOST variables. Users are then logged in under the user name specified and they access the server at the entry point specified in this particular entry.

Default: <*scope*>::HOSTS = * * anonymous /

Example: <*scope*>::HOSTS = pc45.cybermedia.com * user1 welcome

<*scope*>::HTML_DIR: Specifies the directory where PLACE template files and images for WaveMaster layout are stored.

Default: <*scope*>::HTML_DIR = $HOME/wavemaster

<*scope*>::HTTP_TIMEOUT: The time in seconds a connection can be stalled. After that period of time the connection is considered broken. If a connection has the KEEP_ALIVE option set, this timeout also defines the maximum time between user requests after which the connection is closed.

Default: <*scope*>::HTTP_TIMEOUT = 100

<*scope*>::HYPER_HOSTNAME: WaveMaster need not run on the same machine as the low level database engine. In this case this variable tells WaveMaster on which host the database engine is running.

Default: <*scope*>::HYPER_HOSTNAME = localhost

<*scope*>::HYPER_PORT: The port the Hyperwave Information Server is listening to.

Default: <*scope*>::HYPER_PORT = 418

<*scope*>::IMAGE_*: Standard inline images needed by WaveMaster. The values of these variables are either path names relative to <*scope*>::HTML_DIR directory or document names of images on the server. Relative path names are preceded by the string gate:, document names are preceded by hyper:.

Note: *The "anchor" icons described below are necessary for displaying anchors in a list of search results. Because anchors can be given attributes (e.g. title, keyword) they can be found like any other object. By default the anchor icons are the same as the icon of the corresponding object.*

- <*scope*>::IMAGE_CGI: The icon used for displaying a CGI script in a collection listing.

  Default: <*scope*>::IMAGE_CGI = gate:icons/cgi.gif

- <*scope*>::IMAGE_CGI_ANCHOR: The icon used for displaying a link to a CGI script in a list of search results.

  Default: <*scope*>::IMAGE_CGI_ANCHOR = gate:icons/cgianchor.gif

- <*scope*>::IMAGE_CLUSTER: The icon used for displaying a cluster in a collection listing.

  Default: <*scope*>::IMAGE_CLUSTER = gate:icons/cluster.gif

- <*scope*>::IMAGE_CLUSTER_ANCHOR: The icon used for displaying a link to a cluster in a list of search results.

  Default: <*scope*>::IMAGE_CLUSTER_ANCHOR = gate:icons/clusteranchor.gif

- <*scope*>::IMAGE_COLLECTION: The icon used for displaying a collection in a collection listing.

  Default: <*scope*>::IMAGE_COLLECTION = gate:icons/collection.gif

- <*scope*>::IMAGE_COLLECTION_ANCHOR: The icon used for displaying a link to a collection in a list of search results.

  Default: <*scope*>::IMAGE_COLLECTION_ANCHOR = gate:icons/collectionanchor.gif

- <*scope*>::IMAGE_FTP: The icon used for displaying an FTP object in a collection listing.

  Default: <*scope*>::IMAGE_FTP = gate:icons/ftp.gif

- <*scope*>::IMAGE_FTP_ANCHOR: The icon used for displaying a link to an FTP object in a list of search results.

  Default: <*scope*>::IMAGE_FTP_ANCHOR = gate:icons/ftpanchor.gif

- <*scope*>::IMAGE_GENERIC: The icon used for displaying a generic object in a collection listing.

  Default: <*scope*>::IMAGE_GENERIC = gate:icons/generic.gif

- <*scope*>::IMAGE_GENERIC_ANCHOR: The icon used for displaying a link to a generic object in a list of search results.

  Default: <*scope*>::IMAGE_GENERIC_ANCHOR = gate:icons/genericanchor.gif

- <*scope*>::IMAGE_GOPHER: The icon used for displaying a remote Gopher object in a collection listing.

  Default: <*scope*>::IMAGE_GOPHER = gate:icons/gopher.gif

- <*scope*>::IMAGE_GOPHER_ANCHOR: The icon used for displaying a link to a remote Gopher object in a list of search results.

  Default: <*scope*>::IMAGE_GOPHER_ANCHOR = gate:icons/gopheranchor.gif

- <*scope*>::IMAGE_GROUP: The icon used to represent a group.

  Default: <*scope*>::IMAGE_GROUP = gate:icons/group.gif

- <*scope*>::IMAGE_HGICOLL: The icon used for displaying an HGI collection in a collection listing.

  Default: <*scope*>::IMAGE_HGICOLL = gate:icons/hgicoll.gif

- <*scope*>::IMAGE_HGICOLL_ANCHOR: The icon used for displaying a link to a HGI collection in a list of search results.

  Default: <*scope*>::IMAGE_HGICOLL_ANCHOR = gate:icons/hgicollanchor.gif

- <*scope*>::IMAGE_HGIDOC: The icon used for displaying a HGI document in a collection listing.

Default: <*scope*>::IMAGE_HGIDOC = gate:icons/hgidoc.gif

- <*scope*>::IMAGE_HGIDOC_ANCHOR: The icon used for displaying a link to a HGI document in a list of search results.

  Default: <*scope*>::IMAGE_HGIDOC_ANCHOR = gate:icons/hgidocanchor.gif

- <*scope*>::IMAGE_IMAGE: The icon used for displaying an image in a collection listing.

  Default: <*scope*>::IMAGE_IMAGE = gate:icons/image.gif

- <*scope*>::IMAGE_IMAGE_ANCHOR: The icon used for displaying a link to an image in a list of search results.

  Default: <*scope*>::IMAGE_IMAGE_ANCHOR = gate:icons/imageanchor.gif

- <*scope*>::IMAGE_MOVIE: The icon used for displaying a movie in a collection listing.

  Default: <*scope*>::IMAGE_MOVIE = gate:icons/movie.gif

- <*scope*>::IMAGE_MOVIE_ANCHOR: The icon used for displaying a link to a movie in a list of search results.

  Default: <*scope*>::IMAGE_MOVIE_ANCHOR = gate:icons/movieanchor.gif

- <*scope*>::IMAGE_POSTSCRIPT: The icon used for displaying a PostScript document in a collection listing.

  Default: <*scope*>::IMAGE_POSTSCRIPT = gate:icons/postscript.gif

- <*scope*>::IMAGE_POSTSCRIPT_ANCHOR: The icon used for displaying a link to a PostScript document in a list of search results.

  Default: <*scope*>::IMAGE_POSTSCRIPT_ANCHOR = gate:icons/postscriptanchor.gif

- <*scope*>::IMAGE_PROGRAM: The icon used for displaying a program object in a collection listing.

  Default: <*scope*>::IMAGE_PROGRAM = gate:icons/program.gif

- <*scope*>::IMAGE_PROGRAM_ANCHOR: The icon used for displaying a link to a program in a list of search results.

  Default: <*scope*>::IMAGE_PROGRAM_ANCHOR = gate:icons/programanchor.gif

- <*scope*>::IMAGE_SCENE: The icon used for displaying a 3D scene in a collection listing.

  Default: <*scope*>::IMAGE_SCENE = gate:icons/scene.gif

- <*scope*>::IMAGE_SCENE_ANCHOR: The icon used for displaying a link to a 3D scene in a list of search results.

  Default: <*scope*>::IMAGE_SCENE_ANCHOR = gate:icons/sceneanchor.gif

- <*scope*>::IMAGE_TELNET: The icon used for displaying a telnet object in a collection listing.

  Default: <*scope*>::IMAGE_TELNET = gate:icons/telnet.gif

- <*scope*>::IMAGE_TELNET_ANCHOR: The icon used for displaying a link to a telnet object in a list of search results.

  Default: <*scope*>::IMAGE_TELNET_ANCHOR = gate:icons/telnetanchor.gif

- <*scope*>::IMAGE_TEXT: The icon used for displaying a text document in a collection listing.

  Default: <*scope*>::IMAGE_TEXT = gate:icons/text.gif

- <*scope*>::IMAGE_TEXT_ANCHOR: The icon used for displaying a link to a text document in a list of search results.

  Default: <*scope*>::IMAGE_TEXT_ANCHOR = gate:icons/textanchor.gif

- <*scope*>::IMAGE_UNKNOWN: The icon used for displaying a document of unknown type in a collection listing.

  Default: <*scope*>::IMAGE_UNKNOWN = gate:icons/unknown.gif

- <*scope*>::IMAGE_UNKNOWN_ANCHOR: The icon used for displaying a link to a document of unknown type in a list of search results.

  Default: <*scope*>::IMAGE_UNKNOWN_ANCHOR = gate:icons/unknownanchor.gif

- <*scope*>::IMAGE_USER: The icon used to represent a user.

  Default: <*scope*>::IMAGE_USER = gate:icons/user.gif

- <*scope*>::IMAGE_WAIS: The icon used for displaying a WAIS object in a collection listing.

  Default: <*scope*>::IMAGE_WAIS = gate:icons/wais.gif

- <*scope*>::IMAGE_WAIS_ANCHOR: The icon used for displaying a link to a WAIS object in a list of search results.

  Default: <*scope*>::IMAGE_WAIS_ANCHOR = gate:icons/waisanchor.gif

- <*scope*>::IMAGE_WWW: The icon used for displaying a remote WWW object in a collection listing.

  Default: <*scope*>::IMAGE_WWW = gate:icons/www.gif

- <*scope*>::IMAGE_WWW_ANCHOR: The icon used for displaying a link to a remote WWW object in a list of search results.

  Default: <*scope*>::IMAGE_WWW_ANCHOR = gate:icons/wwwanchor.gif

- <*scope*>::IMAGE_WWW_SECURE

  Default: <*scope*>::IMAGE_WWW_SECURE = gate:icons/wwwsecure.gif

- <*scope*>::IMAGE_WWW_SECURE_ANCHOR

  Default: <*scope*>::IMAGE_WWW_SECURE_ANCHOR = gate:icons/wwwsecureanchor.gif

<*scope*>::JAVASCRIPT_FILE_READ_ACCESS: This variable lets you give JavaScript in the Hyperwave templates read access to a directory including its recursive subdirectories. You can make multiple entries in order to specify multiple directories. The value is a full directory path. Default value is the `wavemaster` directory. Note that if this variable is set, JavaScript will no longer have access to the default directory (access to it must be set explicitly).

<*scope*>::JAVASCRIPT_FILE_WRITE_ACCESS: This variable lets you give JavaScript in the Hyperwave templates write access to a directory including its recursive subdirectories. You can make multiple entries in order to specify multiple directories. The value is a full directory path. Default value is the `wavemaster` directory. Note that if this variable is set, JavaScript will no longer have access to the default directory (access to it must be set explicitly).

<*scope*>::JAVASCRIPT_MAX_MEMORY: This variable lets you configure the maximum amount of memory used by the JavaScript runtime engine for server-side JavaScript. Its value is in

bytes and the default value is 1,000,000 bytes. In most cases you will not need to change the value of this variable.

<*scope*>::JAVASCRIPT_STACK_SIZE: This variable is used to configure the size of the stack for the execution of server-side JavaScript. Its value is in bytes and the default value is 8000 bytes. In most cases you will not need to change the value of this variable.

<*scope*>::KEEP_ALIVE: Defines whether the HTTP keep alive mechanism should be activated. Be careful when using keep alive with Netscape 2.02 or lower as it can end in odd results.

Default: <*scope*>::KEEP_ALIVE = TRUE

<*scope*>::LOG: The path to WaveMaster's access log file. Normal logging can be turned off using the off parameter (<*scope*>::LOG[off]).

Default: <*scope*>::LOG = $HOME/log/wave.log

<*scope*>::LOG_FLUSH: Specifies after how many output lines the access log file should be flushed.

Default: <*scope*>::LOG_FLUSH = 100

<*scope*>::LOG_COMMON: It is possible to have WaveMaster write a log file in Common Log File Format. This variable lets you specify a name for the file. The normal WaveMaster log file is produced by default, and this file can be produced instead of it or additionally.

<*scope*>::LOG_COMMON_FLUSH: Specifies after how many output lines the common log file should be flushed.

<*scope*>::LOG_COMBINED: It is possible to have WaveMaster write a log file in Combined Log File Format. This variable lets you specify a name for the file. The normal WaveMaster log file is produced by default, and this file can be produced instead of it or additionally.

<*scope*>::LOG_COMBINED_FLUSH: Specifies after how many output lines the combined log file should be flushed.

<*scope*>::MAGIC_FILE: Tells WaveMaster where to find the magic number file.

Default: <*scope*>::MAGIC_FILE = $HOME/wavemaster/idx/magic

<*scope*>::PORT: Specifies the port WaveMaster is listening to.

Default: <*scope*>::PORT = 80

<*scope*>::PREFERRED_MIMETYPES: The value of this variable is used in conjunction with alternative clusters. If the user does not have preferred mime types entered in his or her user record, the order set with this variable is used. It is also possible to set the default quality with this variable e.g. with a value such as (text,image;Quality=10).

Default: <*scope*>::PREFERRED_MIMETYPES =
(text,image,audio,application/postscript,x-world,movie)

<*scope*>::REQUEST_TIMEOUT: The value of this variable is a time in seconds. If a request (from waveslave) is not processed within this time, it is interrupted and an error message is displayed.

Default: <*scope*>::REQUEST_TIMEOUT = 120

<*scope*>::ROBOTS_FILE: This file contains information about which clients are considered robots rather than standard clients. The order of client/robot detection is to first look at the CLIENTS_FILE to determine if it is a client, if it cannot be found there look at the ROBOTS_FILE to see if it is a robot. If the client cannot be found in either file it is considered a standard client. This file is generated automatically and should not be changed.

Default: <*scope*>::ROBOTS_FILE = $HOME/wavemaster/idx/robots.idx

<*scope*>::SSL_CERTIFICATE: This tells WaveMaster where to find the Privacy Enhanced Mail (PEM) certificate. The PEM certificate does not come with the server.

Default: <*scope*>::SSL_CERTIFICATE = $HOME/server.pem

<*scope*>::SLAVEPORT: Defines the port for the WaveSlave. If not defined then default is 4718. See Starting a second WaveMaster for the NT Hyperwave Information server.

<*scope*>::TEMPLATE_MASTER: Specifies the master template file (written in PLACE) to be used.

Default: <*scope*>::TEMPLATE_MASTER = $HOME/wavemaster/master.html

<*scope*>::USERMAPPING: This variable is used to configure user mapping on the server. User mapping makes it possible to automatically log in users under a particular name according to the host computer they are using to access Hyperwave. See page 65 for more details.

<*scope*>::USER_PLACE_TEMPLATES: This variable is used to turn user templates on or off for all users. Values can be `true` or `false`. Default is `false`.

<*scope*>::USER_PLACE_TEMPLATE_CACHE_ELEMENTS: User templates are cached in each waveslave session. This variable specifies the number of different templates which can be cached at once in each waveslave. If the cache is full and a new template is to be inserted, an old one is removed from the cache (and reevaluated later if needed).

Default: <*scope*>::USER_PLACE_TEMPLATE_CACHE_ELEMENTS = 5

<*scope*>::USER_PLACE_TEMPLATE_CACHE_TTL: User templates are cached in each waveslave session. This variable specifies the time in seconds a user template can live in the cache. After that period of time the user template is removed from the cache (and reevaluated later if needed).

Default: <*scope*>::USER_PLACE_TEMPLATE_CACHE_TTL = 600

<*scope*>::USER_TEMPLATES_GROUPS: This variable is used to turn user templates on or off for the specified user groups. Its value is a list of Hyperwave group names separated by commas.

<*scope*>::USE_COOKIES: This variable is used to turn cookies on or off for anonymous users. This turns off all cookies except the session key cookie because it is necessary if the user wants to log in to the server. Note that when cookies are turned off, no user preferences can be stored and the collection view may not work properly.

Default: <*scope*>::USE_COOKIES = true

<*scope*>::USE_THREADS: This variable is used to switch multithreading on and off in the WaveMaster. Multithreading is available for the Windows NT, DEC Alpha and Sun Solaris platforms, and can greatly decrease the memory consumption of the server if used. Note that this variable can only be used in the UNIX versions of the server for which multithreading is available, as multithreading for Windows NT is always turned on. Turning off multithreading can, for example, be useful in order to see (e.g. with `ps`), how many WWW sessions are active in order to optimize a session pool.

Default: <*scope*>::USE_THREADS = TRUE

2.2.6.1 STARTING A SECOND WAVEMASTER FOR THE NT HYPERWAVE INFORMATION SERVER

You can start a second WaveMaster which uses templates other than the default ones with lines like the following:

```
MAIN::PROCESS = WAVEMASTER
WAVEMASTER::COMMAND = wavemaster -scope WAVEMASTER
WAVEMASTER::HOSTNAME = fiicm1pc86.tu-graz.ac.at
WAVEMASTER::PORT = 80
WAVEMASTER::SLAVEPORT = 4720
WAVEMASTER::DIR = $HOME/wavemaster
WAVEMASTER::LOG_COMMON = $HOME/log/wave.log
WAVEMASTER::ERRORS[off]
WAVEMASTER::LOG[off]
```

```
WAVEMASTER::DEFAULT_LANGUAGE = ge
WAVEMASTER::CACHE_CONTROL = false
WAVEMASTER::DEFAULT_POOLENTRY = default 1 5 1 5

MAIN::PROCESS = WAVESLAVE
WAVESLAVE::COMMAND = waveslave.exe WaveMaster WAVEMASTER "$HOME/.db.contr.rc"
WAVESLAVE::DIR = $HOME/WaveMaster

MAIN::PROCESS = WM-RUV
WM-RUV::COMMAND = wavemaster -scope WM-RUV
WM-RUV::HOSTNAME = fiicm1pc86.tu-graz.ac.at
WM-RUV::PORT = 88
WM-RUV::SLAVEPORT = 4721
WM-RUV::HTML_DIR = $HOME/wavemaster-ruv
WM-RUV::LOG_COMMON = $HOME/log/wave_ruv.log
WM-RUV::ERRORS[off]
WM-RUV::LOG[off]
WM-RUV::DEFAULT_LANGUAGE = ge
WM-RUV::CACHE_CONTROL = false
WM-RUV::DEFAULT_POOLENTRY = default 1 5 1 5

MAIN::PROCESS = WAVESLAVE-RUV
WAVESLAVE-RUV::COMMAND = waveslave.exe WaveMaster WM-RUV "$HOME/.db.contr.rc"
WAVESLAVE-RUV::DIR = $HOME/WaveMaster
```

2.2.7 CONFIGURING WAVESETUP

Even WaveSetup, Hyperwave's GUI configuration tool, can be configured in `.db.contr.rc`.

WaveSetup normally reads the identification from the UNIX account where it is started. If you are unable to use that (on Windows NT, for example), you may solve this by adding the following lines to the configuration file:

`MAIN::USER=` *user_name*

`MAIN::PASSWD=` *encrypted_passwd*

Note: *The variables above were formerly called* `WAVESETUP::USER` *and* `WAVESETUP::PASSWD`*. This changed in Hyperwave version 2.1.*

Below is a complete list of variables which are used to configure WaveSetup.

`WAVESETUP::PORT`: The port WaveSetup listens to. Default is 9999. This entry is overridden by the `-port` command line option.

`WAVESETUP::ALLOWED_HOST`: A domain name address from which connections to WaveSetup are allowed. There may be more than one `ALLOWED_HOST` entry. The value need not be a whole domain name (for example `.hyperwave.com` may be used to allow access from all hosts in the domain hyperwave.com). Connections from localhost are always allowed.

`WAVESETUP::ALLOWED_IP`: An IP address from which connections to WaveSetup are allowed. There may be more then one `ALLOWED_IP` entry. Connections from localhost are always allowed.

Note: *By default, only connections from localhost are allowed. If connections from other computers should be allowed, you must use* `WAVESETUP::ALLOWED_HOST` *and* `WAVESETUP::ALLOWED_IP` *to configure this. This is also necessary if the browser connects to WaveSetup over a proxy.*

The following parameters with foreign scopes are also read by WaveSetup:

`MAIN::USER`: Name of the user allowed to see/change server settings. If no user is specified the user currently running WaveSetup (normally `hwsystem`) is used. This entry is also read by `hwservercontrol`.

MAIN::PASSWD: The encrypted password needed to see/change server settings. If no password is specified the encrypted password of the user currently running WaveSetup (normally hwsystem) is used. This entry is also read by hwservercontrol.

WAVEMASTER::HOSTNAME: The official host name as used by WaveMaster.

2.2.8 CONFIGURING HWBACKUP

HWBackup is the process used to make backups of the Hyperwave Information Server for Windows NT. The .db.contr.rc variables used to configure the process are explained below.

MAIN::TIMEDPROCESS = HWBACKUP

This variable starts the HWBackup process in a timed fashion. This is done in conjunction with

HWBACKUP::START[*interval*] = *start_time*

The *interval* tells the control program how often to start the backup. It may be either *once*, *hourly*, *daily*, *weekly*, *monthly*, *mo* (start every Monday), *tu*, *we*, *th*, *fr*, *sa*, *su* or any (numeric) interval specified in minutes. You may specify more than one period in a comma-separated list (for example use *mo,tu,we,th,fr* to start backup every working day. If no parameter is specified the default is daily. The *start_time* tells the control program when (in GMT!) to start backup for the first time. The format is the usual Hyperwave date string ([yy]yy/mm/dd hh:mm:ss); if the date part is missing it is assumed to be today. If no start time is specified, the time of the last server start is used.

Example: To start backup every Sunday at 1:00 starting at April 1, 1998 use:

```
HWBACKUP::START[su] = 1998/04/01 01:00
```

HWBACKUP::COMMAND: This is like the COMMAND variables for all processes started with the control program. For backup, its value is hwbackup –backup.

Below are further variables that can be configured for backup.

HWBACKUP::BACKUPCOMMAND: The command for the backup process. The value is hwbackupcopy by default. It can be changed to any program which can copy files, e.g. pkzip.

HWBACKUP::BACKUPCOMMANDLINE: This variable is used to configure the command line for the backup process using placeholders. By default it is <BACKUP_COMMAND> <BACKUP_DESTINATION> <FILES_TO_BACKUP> and uses the values of the HWBACKUP::BACKUPCOMMAND and HWBACKUP::DESTINATIONDIR respectively. <FILES_TO_BACKUP> cannot be changed.

HWBACKUP::DESTINATIONDIR: The directory where the backup is stored.

HWBACKUP::MAILTO: The backup process sends a notification mail to this address.

HWBACKUP::LOG: The log file for the backup process.

HWBACKUP::RESTORECOMMANDLINE: This variable is used to configure the command line for the restore process using placeholders. By default it is <RESTORE_COMMAND> <RESTORE_SOURCE> $HOME and uses the values of the HWBACKUP::RESTORECOMMAND and HWBACKUP::DESTINATIONDIR respectively.

HWBACKUP::RESTORECOMMAND: The command for the restore process, e.g. pkunzip if you used pkzip as backup command.

See also page 52 for an explanation of configuring server backup for Windows NT.

2.2.9 CONFIGURING WAVENOTIFY

The `wavenotify` process periodically starts searches on the server using query objects created by users and sends the respective users the results of the search by email.

These automatic searches that users can configure in Hyperwave using the WaveMaster interface are called *subscriptions*. See the *Hyperwave User's Guide* to find out how to create query objects and subscriptions.

ENABLING SUBSCRIPTIONS

`wavenotify`, query objects and subscriptions are a feature of Hyperwave starting with version 2.6. Note that the `wavenotify` process must be running for subscriptions to work, and by the default configuration, this process is started with the server after making a new installation. However, if you are upgrading to version 2.6 or higher from version 2.5 or lower, the process is not started because your old configurations are still used after update. If this is the case, use WaveSetup, the Hyperwave configuration tool, to enable subscriptions. This is done as follows:

1. Access WaveSetup with a browser with the URL `http://`*<your_server_name>*`:9999`
2. Select WaveSetup's **Searching** tab.
3. Under **Subscriptions**, click on the **Enable subscriptions** checkbox.
4. Click on the **Update Searching Settings** button to make the changes in Hyperwave's configuration file.
5. **Stop and restart your server**. This must be done for the changes to take effect. See the *Hyperwave Installation Guide* for instructions on how to do this.

CONFIGURABLE PARAMETERS

The parameters listed below are configurable for the `wavenotify` process in the file `.db.contr.rc`.

`WAVENOTIFY::LOG`: This variable tells `wavenotify` where to put its log file and tells `hwservercontrol` what to do with old log files when the server is restarted. The value is the name and path of the log file. The default value is `$HOME/log/notify.log`. This variable uses the parameters `remove_old` and `no_rename` (see `FTSERVER::LOG`).

`WAVENOTIFY::SENDER`: The email address that appears in the "from:" field of the emails sent by `wavenotify`. Default is *<account_under_which_the_server_is_running>*`@`*<your_domain>*, e.g. `hwsystem@hyperwave.com`.

`WAVENOTIFY::INTERVAL`: The time interval (in minutes) between searches for query objects.

`WAVENOTIFY::HWHOST` and `WAVENOTIFY::HWPORT`: Host name and port, respectively, of the server.

`WAVENOTIFY::WWWHOST` and `WAVENOTIFY::WWWPORT`: Host and port of Wavemaster, respectively. These two variables are only required when Wavenotify is running on a computer other than that which the server/WaveMaster are running on, or if the server/WaveMaster aren't using the standard port numbers (418/80).

The following parameters with foreign scopes are also read by `wavenotify`:

`MAIN::SMTPHOST` and `MAIN::SMTPPORT`: These variables configure the host and port for the SMTP server (the server used to send emails). Under UNIX this should not have to be configured (the default "localhost:25" should always work). This variable is also read by `hwservercontrol` and `hwbackup`.

ENABLING WAVENOTIFY WITH .DB.CONTR.RC

To enable Wavenotify using `.db.contr.rc`, enter the lines

```
MAIN::PROCESS = WAVENOTIFY
WAVENOTIFY::COMMAND = wavenotify.exe
```

in the file, and stop and restart your server.

2.2.10 CONFIGURING THE LDAP GATEWAY

The LDAP gateway is an external identification gateway. It is configured using the following variables.

`HWLDAPD::PORT`: The TCP port for the external user request from `hgserver`.

Default: `HWLDAPD::PORT = 4568`

`HWLDAPD::LDAPHOSTNAME`: Host name of the LDAP server.

Default: localhost

`HWLDAPD::LDAPPORT`: Port of the LDAP server.

Default: `HWLDAPD::LDAPPORT = 389`

`HWLDAPD::LDAPBINDDN`: The distinguished name (DN) with which the gateway reports to the LDAP server (bind operation).

`HWLDAPD::LDAPBINDPASSWD`: The password which is sent to the server with the LDAPBINDDN.

`HWLDAPD::LDAPBASEDN`: The base DN (the organization for which the LDAP server is responsible and in which the desired users are administrated). This variable is required.

Default: none

`HWLDAPD::LDAPUNAMEFILTER`: The name of the attribute where the users' names are stored. This variable is required.

Default: none

`HWLDAPD::LDAPPASSWDATTR`: The name of the attribute where the users' password(s) are stored. This variable is required.

Default: none

`HWLDAPD::LDAPGROUPATTR`: The name of the attribute where the group(s) are stored. This variable is required.

Default: none

2.3 CONFIGURING THE WAVEMASTER INTERFACE

2.3.1 THE PLACE LANGUAGE

The appearance of WaveMaster's user interface is configured by a special meta-HTML language called PLACE. See the *Hyperwave Programmer's Guide* for details.

2.3.2 CONFIGURING THE TOOLBAR AND MENUS

It is possible to configure various aspects of the menus in the Hyperwave Information Server interface, such as the fonts, colors and images used in the menus. It is also possible to configure the menu structure, the menu item texts and the functions carried out when a given menu item is selected.

2.3.2.1 CONFIGURING COLOR AND STYLE OF MENUS

The color, font, font size, etc. of menu items and the color and images used for menus are configured in a file called `mstyle.js`.

Note: The names of the variables in the `mstyle.js` *file are case-sensitive.*

MBSTYLE

The variables in the file `mstyle.js` that are prefixed with "mbstyle" configure aspects of the menu bar.

`mbstyle.FontFace`

The font used in the menu bar. Several font names can be given as the value of this variable. Going through the list from beginning to end, the first font which is available on a given computer is used.

```
mbstyle.FontFace="Univers 55,MS Sans Serif,Arial,Helvetica";
```

`mbstyle.FontSizeAbs`

The font size in pt.

```
mbstyle.FontSizeAbs="10pt";
```

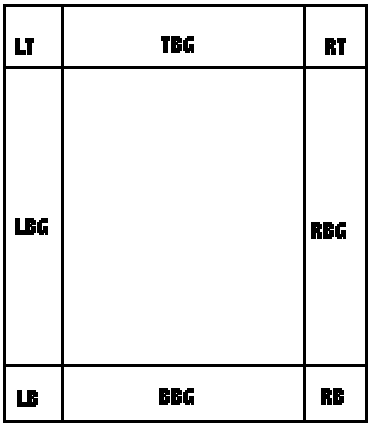
`mbstyle.Decoration`

The toolbar can be configured to use graphics or simply colors. When the value of this variable is "full", graphics are used, and when it is "low", only colors are used.

```
mbstyle.Decoration="full";
```

MENU BAR STRUCTURE

The toolbar and menus are structured as follows:



They can be configured so as to place a graphic in any of the eight labeled spaces on the edge of the menu shown in the figure above. Each corner is labeled according to its location (e.g. "LT" for "left top", "RB" for "right bottom") and these labels are used in the variables described below. The edges are labeled with "BG" for background, e.g. "LBG" is "left background". For the edges, the configured image is actually repeated to fill up the space. It is also possible to superimpose an image on one of the edges, e.g. `mbstyle.MenuTImg` is used to superimpose an image on the top edge of the menu bar.

`mbstyle.MenuTImg`

This causes the given image to appear on the top edge of the menu. Exactly where it appears (left, center or bottom) is configured using `mbstyle.MenuImgAlign`.

`mbstyle.MenuLTImg`

This variable gives the location of the image that should appear in the top left corner of the menu.

```
mbstyle.MenuLTImg="/wavemaster/v4.1/config/ui_styles/judy/mb/upperleft.gif";
```

`mbstyle.MenuTBG`

This gives the image that should appear on the top edge of the menu. This image is duplicated several times to fill up the space.

```
mbstyle.MenuTBG="/wavemaster/v4.1/config/ui_styles/judy/mb/top.gif";
```

`mbstyle.MenuRTImg`

The configures the image that should appear in the top right corner of the menu.

```
mbstyle.MenuRTImg="/wavemaster/v4.1/config/ui_styles/judy/mb/upperright.gif";
```

`mbstyle.MenuLImg`

This causes the given image to appear on the left edge of the menu. Exactly where it appears (left, center or bottom) is configured using `mbstyle.MenuImgAlign`.

```
mbstyle.MenuLImg="/wavemaster/v4.1/config/ui_styles/judy/mb/left.gif";
```

`mbstyle.MenuRImg`

This causes the given image to appear on the right edge of the menu. Exactly where it appears (left, center or bottom) is configured using `mbstyle.MenuImgAlign`.

```
mbstyle.MenuRImg="/wavemaster/v4.1/config/ui_styles/judy/mb/right.gif";
```

`mbstyle.MenuLBImg`

The image that should appear in the left bottom corner of the menu.

```
mbstyle.MenuLBImg="/wavemaster/v4.1/config/ui_styles/judy/mb/lowerleft.gif";
```

`mbstyle.MenuRBImg`

The image that should appear in the right bottom corner of the menu.

```
mbstyle.MenuRBImg="/wavemaster/v4.1/config/ui_styles/judy/mb/lowerright.gif";
```

`mbstyle.MenuBBG`

The image that should appear on the bottom edge of the menu. This image is duplicated several times to fill up the space.

```
mbstyle.MenuBBG="/wavemaster/v4.1/config/ui_styles/judy/mb/bottom.gif";
```

`mbstyle.MenuBGImg`

The image that should appear as the background of the menu.

```
mbstyle.MenuBGImg="/wavemaster/v4.1/config/ui_styles/judy/mb/backidle.gif";
```

`mbstyle.BImg`

This causes the given image to appear on the bottom edge of the menu. Exactly where it appears (left, center or bottom) is configured using `mbstyle.MenuImgAlign`.

`mbstyle.MenuImgAlign`

This variable is used to configure the position of a graphic placed on one of the edges of the menu by e.g. `mbstyle.MenuRImg`. It determines whether the graphic appears on the top, center or bottom of the edge. It uses a parameter depending on which edge the graphic is being configured

for (R,L,T or B) and its value can be "left", "center" or "bottom". For example, to place a graphic on the bottom of the right edge of the menu, you must put lines like the following in your file:

```
mbstyle.MenuRImg="/wavemaster/v4.1/config/ui_styles/judy/mb/rig
ht.gif";
mbstyle.MenuImgAlign[R]=bottom;
mbstyle.Opener
```

This image is displayed when the menu bar is closed. Clicking on it opens the menu bar. The opener can be seen in Figure 3 below the Hyperwave graphic.

```
mbstyle.Opener="/wavemaster/v4.1/config/ui_styles/judy/mb/opene
r.gif";
```

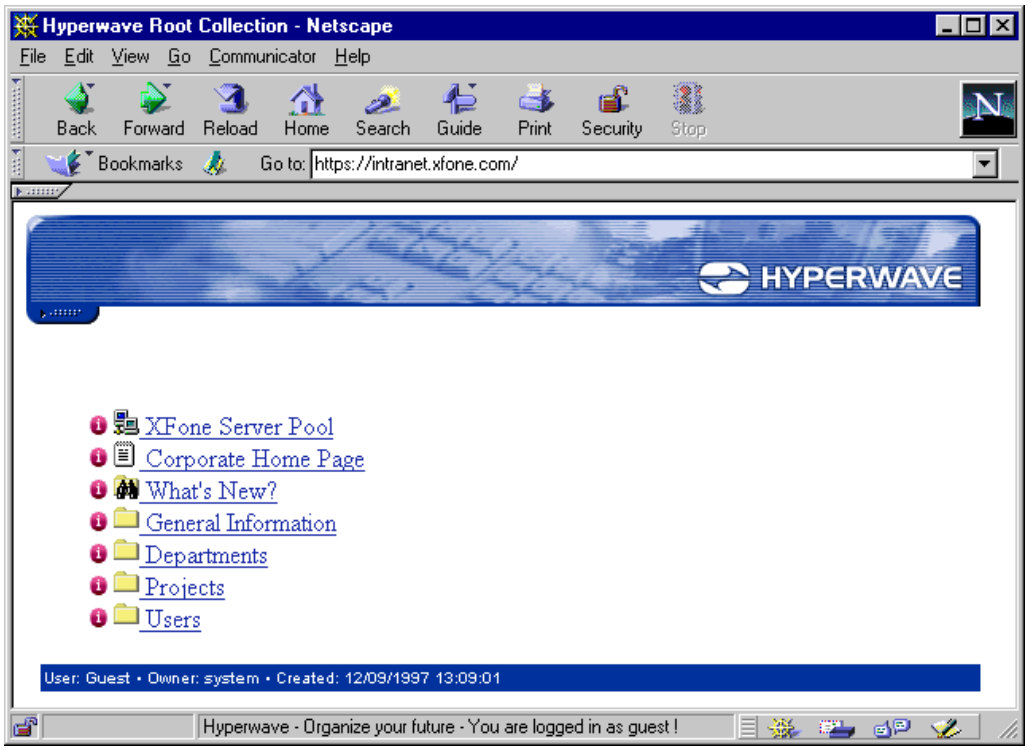


Figure 3: The menu bar opener

mbstyle.Handle

This image is displayed on the left side of the opened menu bar (see Figure 4) and is used to close and move the menu bar.

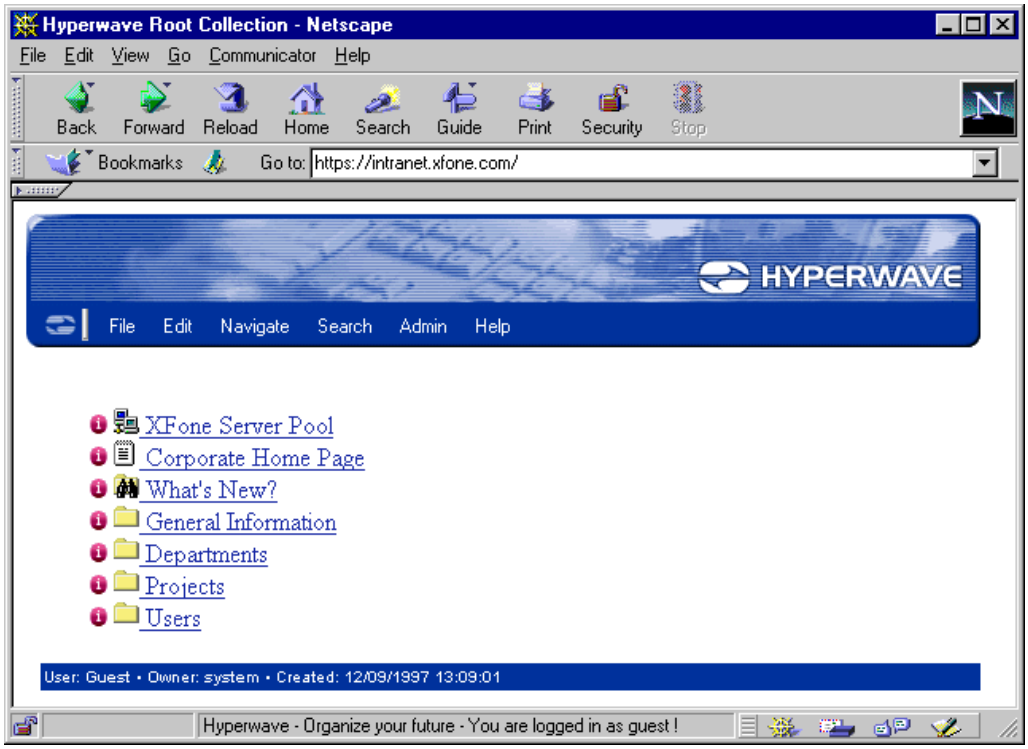


Figure 4: The menu bar handle

`mbstyle.HandleInv`

The image which is used for the inverted handle. The inverted handle is displayed when a user places the mouse pointer over the handle.

GIVING THE TOOLBAR A 3D APPEARANCE

`mbstyle.BorderColor1`

`mbstyle.BorderColor2`

`mbstyle.BorderColor3`

The three variables listed above are used to give menus a 3-dimensional appearance. As general guidelines, `mbstyle.BorderColor1` should be a very light color, `mbstyle.BorderColor2` should have the same value as `mbstyle.ItemBGColor`, and `mbstyle.BorderColor3` should be very dark. For example:

```
mbstyle.BorderColor1="white";
mbstyle.BorderColor2="#0e4793";
mbstyle.BorderColor3="black";
```

`mbstyle.ItemSelColor`

The text color of the currently selected menu item.

```
mbstyle.ItemSelColor="black";
```

`mbstyle.ItemColor`

The color used for the text of menu items.

```
mbstyle.ItemColor="#ffffff";
```

`mbstyle.ItemBGColor`

The background color of menu items.

```
mbstyle.ItemBGColor="#0e4793";
```

`mbstyle.ItemSelBGColor`

The background color of the currently selected menu item.

```
mbstyle.ItemSelBGColor="#ff9900";
mbstyle.CheckedIcon
mbstyle.CheckedIconInv
mbstyle.ChildTIcon
mbstyle.ChildTIconInv
mbstyle.ChildLIcon
mbstyle.ChildLIconInv
mbstyle.ChildIconWidth
mbstyle.CheckedIconWidth
mbstyle.IconWidth
```

LISTYLE Variables prefixed with "listyle" configure aspects of the collection listing.

```
listyle.headBGColor
listyle.headBGColor = "white";
listyle.attribColor
```

This is the color of the attributes (e.g. version number, author name) of items in the collection listing.

```
listyle.attribColor="#999999";
```

HSTYLE Variables prefixed with "hstyle" configure aspects of the graphic which is placed above the menu bar.

```
hstyle.BGImg
```

The image which appears above the menu bar.

```
hstyle.BGImg="/wavemaster/v4.1/config/ui_styles/icons/header.gi
f";
```

DSTYLE Variables prefixed with "dstyle" configure aspects of dialogs.

```
dstyle.BGColor
```

The background color of dialogs.

```
dstyle.BGColor="#DEDEDE";
dstyle.Color
```

The color of fonts used in dialogs.

```
dstyle.Color="black";
dstyle.FontFace="Arial, Geneva, Helvetica";
```

The type of font used in dialogs.

```
dstyle.decoration="full";
```

SUB-STYLES OF DSTYLE The "dialog" style is divided into several sub-styles, such as "Caption", "Desc", etc. which each configure a particular style within a dialog. For each of these styles, font size and color can be configured. These settings override the corresponding "dbstyle" settings.

DSTYLE.CAPTION

```
dstyle.CaptionFontSize
```

Font size for the "Caption" style.

```
dstyle.CaptionFontSize="18pt";
```

DSTYLE.CAPTIONC

```
dstyle.CaptionCFontSize
```

Font size for the "CaptionC" style.

```
dstyle.CaptionCFontSize="18pt";
dstyle.CaptionCColor
```

Color for the "CaptionC" style.

```
dstyle.CaptionCColor="#003399";
```

DSTYLE.DESC

```
dstyle.DescFontSize
```

Font size for the "Desc" style.

```
dstyle.DescFontSize="8pt";
```

```
dstyle.DescColor
```

Color for the "Desc" style.

```
dstyle.DescColor="#003399";
```

DSTYLE.ATTRVALUE

```
dstyle.AttrValueColor
```

Color for the "AttrValue" style.

```
dstyle.AttrValueFontSize
```

Font size for the "AttrValue" style.

DSTYLE.ATTRNAME

```
dstyle.AttrNameFontSize
```

The font size for the "AttrName" style.

DSTYLE.OBJLIST

```
dstyle.ObjListFontSize
```

The font size for the "ObjList" style.

DSTYLE.OBJLISTC

```
dstyle.ObjListCColor
```

The color of the "ObjListC" style.

```
dstyle.ObjListCFontSize
```

The font size of the "ObjListC" style.

DSTYLE.MENU

```
dstyle.MenuLTImg
dstyle.MenuTBG
dstyle.MenuTImg
dstyle.MenuRTImg
dstyle.MenuLImg
dstyle.MenuLBG
dstyle.MenuRImg
dstyle.MenuRBG
dstyle.MenuLBImg
dstyle.MenuRBImg
dstyle.MenuBBG
dstyle.MenuBGImg
dstyle.MenuBImg
```

2.3.2.2 CONFIGURING TEXT AND FUNCTION OF MENU ITEMS

The menu item texts and functions are configured in the file `menu.html`, which is found in `Program Files/Hyperwave/wavemaster/v4.1/config` under Windows NT and in `~hwsystem/wavemaster/v4.1/config` under UNIX. Each menu is created and configured as a separate object.

CREATING MENUS

A menu is created using a line like the one below. The new menu is given a name, in this case "newM", which is used later on when adding items to the menu. The command "new" creates a new instance of the class "Menu". Further, when a menu is created, the configuration it is to use must be given as a list of parameter-value pairs separated by commas and enclosed in brackets:

```
newM=new Menu("HWNewMenu",{menuconf:mitemst,decoration:deco});
```

MENUCONF

The command above produces a new menu called "newM". There are two parameters given for this menu. The value of `menuconf` tells the menu which menu style to use. In this case the value is `mitemst`, which means "menu item style." This means that the new menu takes the values for menu item configuration from the file `mstyle.js`, i.e the variables that are prefixed with "mitstyle." are used. Another possible value for `menuconf` is `mstyle`, which means that the values for configuring the toolbar are used for the new menu, i.e. the parameters that are prefixed with "mbstyle.".

`menuconf` can be given other values as well. For example, you can create your own style and call it, e.g. "mystyle". To do this you would configure all the parameters which exist for e.g. mitstyle, only with the prefix "mystyle". Then you could give the new menu your personal configuration by giving `menuconf` the value `mystyle`.

DECORATION

The second parameter given in this case is `decoration`. This parameter tells the menu which part of the given configuration to use. For each configuration there is a "full" configuration, i.e. one that uses images, and a "low" configuration, i.e. one that uses only colors. If you use one of these values, the menu will always use the chosen decoration. A third possible value for `decoration`, as in this case, is `deco`. This means that the menu uses the decoration which corresponds to the currently selected user preference, which is stored in the session variable `HWMENUSTYLE`.

Now that the menu has been created and its overall configuration established, it is time to add the menu item texts and functions. This is done using the name of the new menu with the extension "add".

```
newM.add("Collection...",{url:"%%action.call(insertcollection.action)%%&Paramet
er=Collection"%%if object.get_attrib(DocumentType) != "collection" ||
!object.is_editable%%,isdisabled:true%%endif%%});
```

As can be seen above, the text for the menu item, "Collection…", is listed first in quotes. It is followed by a comma, and then the menu item function in brackets. This menu item is chosen when a new collection is to be created, and thus it should, of course, bring up the "New Collection" dialog. The parameter "url" is used to send a request to the server to display the "Insert Collection" dialog. As can be seen, the self-defined function `insertcollection.action` is called with the parameter "Collection". The second PLACE statement is an "if" statement that determines whether the menu item should be enabled or not, depending on the value of the **DocumentType** attribute of the current object and whether or not the current object can be edited by the current user. Obviously, if the current object is not a collection, it makes no sense to try to insert a new collection into it. Likewise, if the user has no rights to edit the current object, he or she should not be allowed to insert a collection into it.

Menu items for inserting the other types of collection can be added with similar lines, e.g. to add a cluster, the following entry should be made:

```
newM.add("Cluster...",{url:"%%action.call(insertcollection.action)%%&Parameter=
Cluster"%%if object.get_attrib(DocumentType) != "collection" ||
!object.is_editable%%,isdisabled:true%%endif%%});
```

As can be seen, the Parameter in this case is "Cluster" instead of "Collection".

Similar entries can be made for the remaining collection types Sequence, Alternative Cluster and Multi Cluster.

It is also possible to add a separator to separate items in the menu with a line like:

```
newM.addSep();
```

A separator is used in this particular menu to separate the "new collection" items from the "new document" items.

Below the separator, the first item is used for inserting a new text into the server:

```
newM.add("Text...",{url:"%%action.call(inserttext.action)%%"%%if
object.get_attrib(DocumentType) != "collection" ||
!object.is_editable%%,isdisabled:true%%endif%%});
```

As can be seen in the lines above, this menu item calls the action `inserttext.action` instead of `insertcollection.action`.

In the case of this menu, there are further items for inserting HTML, files, remote documents, and CGI scripts.

After a further separator, a menu item for inserting annotations is added:

```
newM.add("%%ge:Anmerkung,:Annotation%%",{exec:"if (typeof
openAnnWizard=='function')openAnnWizard();else alert ('Annotations not
supported on this page!')"})
```

Because Hyperwave uses a multilingual (German/English) interface, the menu items must appear according to the user's preferred language. In the case of the "Annotation" item, the German text is "Anmerkung", so instead of a simple text as is used in the other menu items, the PLACE construct `%%ge:Anmerkung,:Annotation%%` is used. For the function that this item calls, "exec" is used because the function is executed using JavaScript.

FURTHER MENU PARAMETERS

As can be seen in the example above, the parameters `menuconf` and `decoration` are used to configure the menu. There are several other parameters which can be used to influence menus. Here is a complete list:

`decoration`

Possible values: full, low, deco

Selects the type of decoration used, i.e. "full" for graphics, "low" for colors, "deco" for user-defined. This overrides the common setting.

`FontSize`

Possible values: a font size in pt.

This gives the font size for the menu item texts.

`menuconf`

Possible values: mitemst, mstyle, etc.

menuconf defines which menu configuration is used. With "mitemst" the menu item style is used, with "mstyle" the menu bar style is used. It is also possible to configure other styles.

`obj`

The **Name** attribute of an object on the server. All actions carried out by menu items are carried out on this object. `obj` is used in conjunction with the menu item parameters `act` and `actp`, that is, the values of these parameters are combined and sent to the server.

`orientation`

Possible values: top, left

Default: left

This parameter tells the menu to be either "left-oriented", i.e. a vertical list of menu items, or "top-oriented", i.e. the menu items are placed side by side.

`width`

Possible values: a number which represents a width in pixels

Default: menu is as wide as required for the menu items on it

This parameter allows you to give the menu the desired width.

FURTHER MENU ITEM PARAMETERS

Here is a full list of menu item parameters.

`act`

Possible values: the name of a PLACE action, e.g. `inserttext.action`

The parameter gives the name of a predefined or self-defined PLACE action. The action is carried out with the value of `actp` as parameter if `actp` has been specified.

`actp`

The parameter used with the action defined by `act`.

`child`

Possible values: the name of an existing menu

The menu whose name is given here appears when the menu item is selected.

`exec`

Possible values: JavaScript commands

`icon`

Possible values: the URL of an icon image

This places an icon to the left of the menu text.

`iconinv`

Possible values: the URL of an icon image

This icon appears instead of the icon configured by the last parameter when the mouse pointer is over the menu item.

`ischecked`

Possible values: true, false

Default: false

When true, a check mark appears to the left of themenu item text.

`isdisabled`

Possible values: true, false

Default: false

When true, the menu item is disabled.

`key`

This gives a keyboard shortcut that can be used for reaching the given menu item.

`nohighlite`

Possible values: true, false

Default: false

When true, the menu item is not highlighted when the mouse pointer is placed over it.

`status`

Possible values: a text

This parameter causes the text you enter to appear in the status line at the bottom of the browser when you place the mouse pointer over the menu item.

`target`

Possible values: any name

This names the window that the result of selecting the menu item appears in. This is used to e.g. put all dialogs in the same window instead of opening up a new one for each.

`url`

The URL to be executed when a given menu item is selected.

2.4 CONFIGURING RELEASE PROCEDURES

Certain configurations must be made for Release Procedures (workflow) before it can be used. The file `config.js` contains several parameters which must be configured for Release Procedures to work, and some variables in the server's main configuration file, `.db.contr.rc`, must be configured as well. Also, the manager file and the flow collection must be created. Lastly, the attributes of user records are affected for users who are involved in the workflow process. These configurations are explained below.

CONFIG.JS

The Release Procedures configuration file is found in:

`$Home/wavemaster/v4.1/include/reflow/code/config.js`

THE MANAGER FILE

The manager file is the file where login information for the privileged release manager is stored. The release manager creates and manipulates documents during the flow process and must be a member of the group system. The release manager is necessary because there are steps in Release Procedures in which a document must be checked in or its rights must be changed, but the current user does not have the appropriate rights to do so. For example, when passing a document to the next auditor, the rights of the document must be changed such that the next auditor is given read rights to it, but because the auditor does not have write rights to the document, he cannot do this. However, using the information in the manager file, when the auditor passes on the document, the server logs starts a user session with the release managers account, changes the rights of the object as required, and then ends this session. Note that this process takes place automatically and is completely controlled by the server.

CREATING THE MANAGER FILE

Create a file containing the login information of any system user. The format of the file must be such that the first line contains the server user name of this user and the second line contains the password for this user in plain text.

MAKING THE MANAGER FILE READABLE/SECURITY CONSIDERATIONS

For security reasons, the manager file should be stored in a separate directory that is by default not readable by JavaScript or the templates. The `wavemaster` directory and all of its recursive subdirectories are by default readable by the JavaScript in any Hyperwave template and can be accessed by a respective URL. Thus the file should be located elsewhere. The directory in which it is located, however, must be made readable for Hyperwave Information Server's JavaScript. This is done in the file `.db.contr.rc` using the variable `WAVEMASTER::JAVASCRIPT_FILE_READ_ACCESS`, which must be set to the absolute path of the directory where the manager file is located. Note that if you set the variable `WAVEMASTER::USER_PLACE_TEMPLATES` to "true" (its value is "false" by default), users may write their own PLACE templates, which may contain JavaScript. This JavaScript code has access to what is in the directory specified in `WAVEMASTER::JAVASCRIPT_FILE_READ_ACCESS`. Thus if you want to use user templates, you should restrict the user groups allowed to use such templates to a group of trusted users (such as the system group) with the variable `WAVEMASTER::USER_TEMPLATES_GROUPS`.

PATH TO MANAGER FILE

The variable `releaseconfig.managerfile` in `config.js` tells Release Procedures where the manager file is located. The value of this variable is the path of the manager file, which can be relative to $Home in the Hyperwave Information Server, or be an absolute path to the manager file.

CONFIGURING THE DATABASE HOST

The variable `releaseconfig.HW_databasehost` has as value the Hyperwave Information Server host that WaveMaster is connected to. This host name is also typically set in `.db.contr.rc` using the variable `WAVEMASTER::HYPER_HOSTNAME`, so it should have the same value as that variable. The data base host must be set in both places because JavaScript does not have access to the `.db.contr.rc` variable.

CONFIGURING THE FLOW COLLECTION

The variable `releaseconfig.flowcollection` tells Release Procedures where to find the flow collection. You must create a collection for this purpose on the server and then give this

variable the Name attribute of the collection as value. This is the collection where all release workflow relevant data is stored, i.e. all procedures as well as some status information.

CONFIGURING THE SMTP HOST

The variable `releaseconfig.smtphost` is used to configure the SMTP host (mail gateway host) and has as value the name of this host. If this is not configured properly, no notification mails can be sent.

USERS WHO WORK WITH RELEASE PROCEDURES

To give a user the privilege to create new release procedures, the attribute **ReleaseAdmin** with the value "yes" must be inserted in his or her user record. Note that all system users automatically have the right to create new procedures.

For all users who are involved in the workflow process, whether they are release administrators, publishers or auditors, their user record should contain the attribute **Email** with the respective email address as value. This attribute is used to send release flow notification to users.

2.5 CUSTOM INDEXES

Hyperwave supports indexing of arbitrary attributes. How to configure a custom index is explained below.

2.5.1 CUSTOM INDEX TYPES

The value of a custom index field can be of one of the following types:

- **Keyword**

  This type allows you to enter one or more words separated by spaces. Each word entered is indexed separately.

- **LangKeyword**

  LangKeyword is like **Keyword** except that it is used to associate a language with the entry or entries made. When entering a value of type **LangKeyword** you may enter a valid two-letter language prefix followed by a colon at the beginning of the entry, e.g. `ge:Beispiel`. If the Hyperwave language prefix (e.g. "en:") is omitted, a default language is supplied (default is "en:" for English).

- **NameKey**

  Entries of type **NameKey** must be a single word, unlike **Keyword** entries. If more than one word is entered, only the first word is indexed and the rest are ignored.

- **Time**

  Entries of type **Time** must be in the format `[yy]yy/mm/dd hh:mm:ss [GMT]`. It is possible to only go to the desired accuracy, e.g. `97/03/07` is interpreted as March 7, 1997 at 12:00am.

The characters which may be used when entering the values of custom attributes are the letters from A to Z in upper or lower case, the digits 0 to 9, the underline ("_") and dash ("-").

2.5.2 CREATING CUSTOM INDEXES

WITH WAVESETUP

Configuring custom indexes with Hyperwave's configuration tool is simple. See page 8 for instructions on how to access WaveSetup for your platform then follow the steps below.

1. Click on the "Searching" tab in WaveSetup.
2. Scroll down to the "Custom Index" section (see Figure 5).
3. Type the attribute name of your choice in the "Field name" field.
4. Select the index type from the listbox.
5. Click on the "Add this index field" button.
6. Stop and restart the server.

WITH .DB.CONTR.RC

Custom indexes can also be configured using Hyperwave's main configuration file, `.db.contr.rc`. Add a line like the following for each index desired if you are using the native object database, `wavestore`:

`WAVESTORE::customindex[`*field_name*`] =` *index_type*

where *index_type* is either `Keyword`, `LangKeyword`, `NameKey` or `Time` and *field_name* is the attribute name of your choice.

Here you see three examples for custom indexes:

```
WAVESTORE::customindex[Publisher] = Keyword
WAVESTORE::customindex[Year] = Date
```

The first example indexes the attribute `Publisher` as a keyword, and the second indexes the attribute `Year` as a date.

Note: If you are using an Oracle database instead of `wavestore`*, the scope* `WAVEORACLE` *must be used for these variables*

Note: *The new index file is not created until the next time the server is stopped and restarted.*

AFTER THE INDEX IS CREATED

The server will try to index all corresponding index field values of the objects already contained in the object repository. However, if the syntax of the index field in an object record is not correct (e.g. type **LangKeyword** and an unknown language prefix was specified or type **Time** and the wrong time format), this field will not be indexed.

Each time a new custom index is activated, a new corresponding index file is created. Index files have to be kept open to perform fast random access to the database, therefore each custom index needs its own file descriptor at server runtime.

Note: *Custom attributes are case-sensitive. If, for example, you create the custom attributes "Email" and "email", they will be indexed separately.*

2.5.3 REMOVING CUSTOM INDEXES

WITH WAVESETUP

See page 8 for instructions on how to start WaveSetup for your platform then follow the instructions below.

1. Click on the "Searching" tab in WaveSetup.
2. Scroll down to the "Custom Index" section (see Figure 5).
3. Click on the "Remove" checkbox(es) for the custom index(es) you want to remove.
4. Click on the "Remove checked indexes" button.

5. Stop and restart the server.

WITH .DB.CONTR.RC

To remove a custom index, you first have to stop the Hyperwave Information Server (using the `hwstop` command). Then either delete or comment out the corresponding `WAVESTORE::customindex` directive in the `.db.contr.rc` file. When the server is started again, it will remove the custom index file and no longer index the custom index field.

2.5.4 ATTRIBUTES WHICH CANNOT BE INDEXED

There are a few restrictions when indexing attributes. You cannot configure a custom index for the following attributes:

1. All attributes that are already indexed, e.g. **Title**, **Keyword**, **Name**, etc.
2. The **Account** attribute present in user records and the **License** attribute which is used to control the number of simultaneous users who have access to the documents in a collection are intentionally not indexable for data protection reasons.
3. **CollectionType**, **DocumentType** and **Type** cannot be indexed because they are stored differently than they appear and are converted on the fly into user readable form.
4. **GOid**, **Path** and internal type identifiers such as **TType** cannot be indexed for reliability reasons.

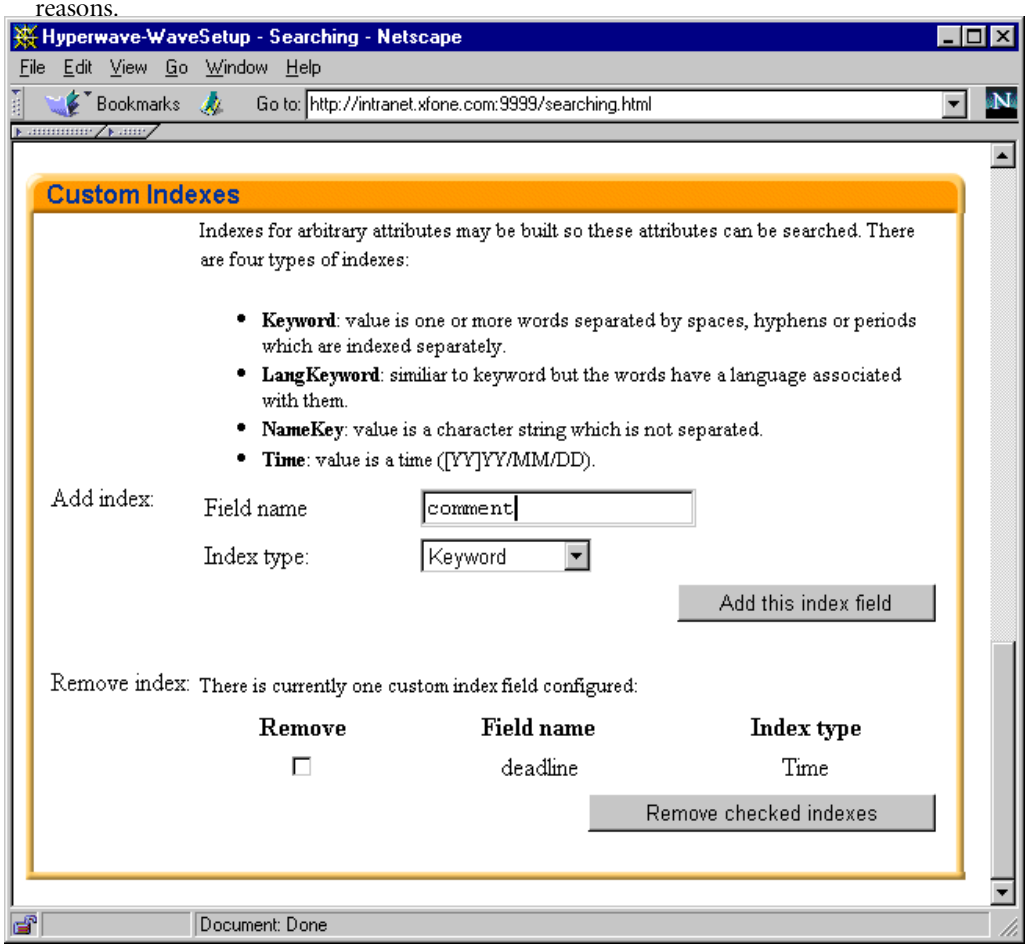


Figure 5: Configuring custom indexes with WaveSetup

2.6 SESSION POOLS

Session pools are a feature of Hyperwave starting with version 2.6. Session pools prevent `hgserver` from starting arbitrarily many sessions under the same user name. This problem can arise when an anonymous user connects to the server with a client that is not using cookies. Every time such a user accesses a document on the server, a new session is started because the server has no way of knowing if the client has already connected and started a session.

WHICH USERS REQUIRE SESSION POOLS

With respect to session pools, actual anonymous users, users logged in to the server through user mapping, and users logged in through multihoming are all considered to be anonymous users. Every time a user of one of these categories logs in, Hyperwave makes sure it gets a session pool if it doesn't already have one. Here there are three cases: either a pool has been specifically configured for the user and was created when the server was started (see below), or no pool has yet been created and the server uses the default pool entry as parameters for a new pool (see below), or the server has already done this because a user with the same name previously logged in anonymously.

Note: Anonymous users as defined above cannot edit any objects on the server, even if the user account they logged in under has such rights in the normal case.

CREATING SESSION POOLS FOR SPECIFIC USERS

It is possible to create session pools for specific user names. This can be done using WaveSetup or by making an entry based on the following guidelines in Hyperwave's main configuration file, `.db.contr.rc` for each session pool desired (where *scope* is the name of the WaveMaster process):

<*scope*>`::ADD_POOLENTRY =` <*user*> <*min*> <*max*> <*startup*> <*timeout*>

The various parameters are defined as follows:

<*user*>: The name of the user the pool entry is for. This user name **must exist** in the Hyperwave user database. If not you will not be able to start the server. An exception to this rule is "anonymous", which always exists in the Hyperwave user database, though not explicitly.

<*min*>: The minimum number of sessions in the pool. If at some time the number of sessions goes below this number, new sessions are added.

<*max*>: The maximum number of sessions in the pool.

<*startup*>: The number of sessions started for the user upon creation of the session pool.

<*timeout*>: The time in seconds a session can be idle before it is closed.

Session pools of this type are started when the server is started and persist the entire time the server is running.

Note: Entries of this type should only be created for users who fall into the anonymous category explained above. If, for example, you create a session pool for the non-anonymous user "smith", other users will be able to log in as smith without entering the password. This could have unpleasant consequences, especially if "smith" has rights to modify objects on the server.

Note: When the server is being started, it may take some time before all configured session pools are started. Users who try to log in under a user name which has a session pool configured for it will not be successful until the pool has been created.

DEFAULT SESSION POOL

If a user of the anonymous type (see above) logs in and no session pool has been started for it because it doesn't have one specifically defined for it, then its session pool is defined by the variable

<*scope*>`::DEFAULT_POOLENTRY = default` <*min*> <*max*> <*startup*> <*timeout*>

where the default value is

<*scope*>`::DEFAULT_POOLENTRY = default 1 5 1 600`

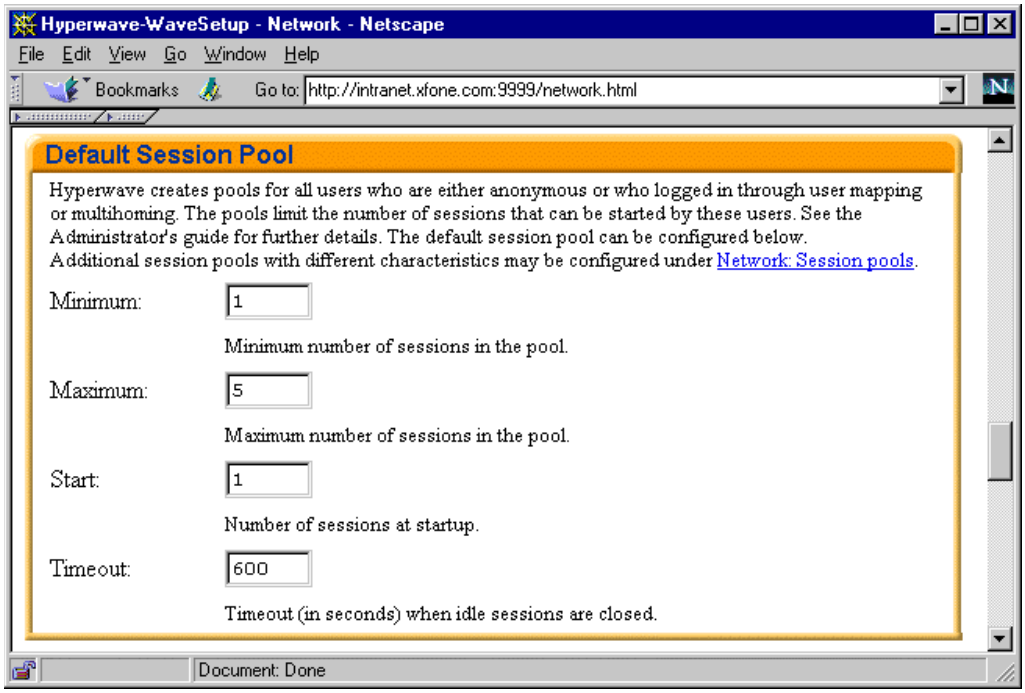


Figure 6: Configuring the default session pool with WaveSetup

CONFIGURING SESSION POOLS WITH WAVESETUP

You can use the WaveSetup configuration tool to configure session pools (see Figure 6).

2.7 EXTENDED LANGUAGE SUPPORT

If you are using the Verity search engine with your server, you can install any languages supported by Verity. By default, the languages English and German are installed. All other languages are available on the Hyperwave CD (Version 2.6) or by ftp (see http://www.hyperwave.de/hw_server/support/verity). How to install a language is explained below.

2.7.1 INSTALLING LANGUAGES

When you install the Hyperwave Information Server, only the locales for English and German are installed. All other locales are available on the CD and are installed as described below.

To install a locale from the Hyperwave CD, you simply need to copy a locale from the `3rdparty/verity/common` directory to the `verity/common` directory in the home directory of the user under which Hyperwave is installed.

2.7.2 SUPPORTED LANGUAGES

There are language locales for the following 10 languages on the Hyperwave CD: Danish, Dutch, English, French, German, Italian, Norwegian, Portuguese, Spanish and Swedish.

2.7.3 FEATURES

Verity offers several advanced fulltext search features.

- **Thesaurus**: Allows you to find documents containing words that have the same meaning as your search term
- **Variations**: Lets you find documents which contai n words which are grammatical variations of your search term.
- **Sounds like**: Lets you find documents containing words that have a similar pronunciation to your search term.

If you enter more than one search term, Verity allows you to specify how you want these words to be associated in the documents found, e.g. in the same sentence, in the same paragraph, in the same document, etc.

2.7.4 RESTRICTIONS

Some features are not available on all platforms.

LinguistX: Linguistics packages based on LinguistX technology from Inxight Software, Inc. The linguistics packages include stemmers, tokenizers, and natural language processing capabilities, features used to support clustering, summarization, query-by-example (free text query parser).

KeyView Filter Kit V1.0: Supports filtering of WYSIWYG documents in numerous formats (see page 18); used for indexing these documents.

Mastersoft Filter Kit V1.5: Supports filtering of WYSIWYG documents in numerous formats; used for indexing these documents.

Platform	LinguistX	KeyView Filtering	Mastersoft Filtering
Windows NT	YES	YES	NO
Sun Solaris	YES	YES	NO
IRIX 5.3	NO	NO	YES
IRIX 6.2	YES	YES	NO
HP-UX	YES	YES	NO
Dec Alpha OSF	YES	YES	NO

2.7.5 A NOTE ABOUT USING NEW LOCALES

If you install a new locale, you will probably want to insert documents into your server and have them indexed as documents of the appropriate language. To do this, you must tell Hyperwave the language of the documents. Currently the forms used when inserting an object in Hyperwave with a Web browser only allow the choices English and German. If you want to insert documents of another language, you must enter a two-letter ISO 639 language prefix followed by a colon before the title. For example, to insert a French document, you might enter `fr:My Title` in the title field. The ISO 639 language prefixes for the languages of the Verity locales available are found in the table below.

Language	Prefix
Danish	da
Dutch	nl
English	en
French	fr
German	de
Italian	it
Norwegian	no
Portuguese	pt
Spanish	es
Swedish	sv

3 MAINTENANCE AND ADMINISTRATION

This chapter describes various tools available for maintaining, administrating and extending the functionality of your server. Topics such as making a backup of the server, creating user accounts, setting up a Hyperwave server pool, using WaveMaster with SSL, etc., are handled here.

3.1 MAKING A BACKUP OF YOUR SERVER

As administrator of a Hyperwave Information Server, you will, of course, want to be able to make a backup of the contents of your server. This section explains the backup procedure for the NT and UNIX platforms.

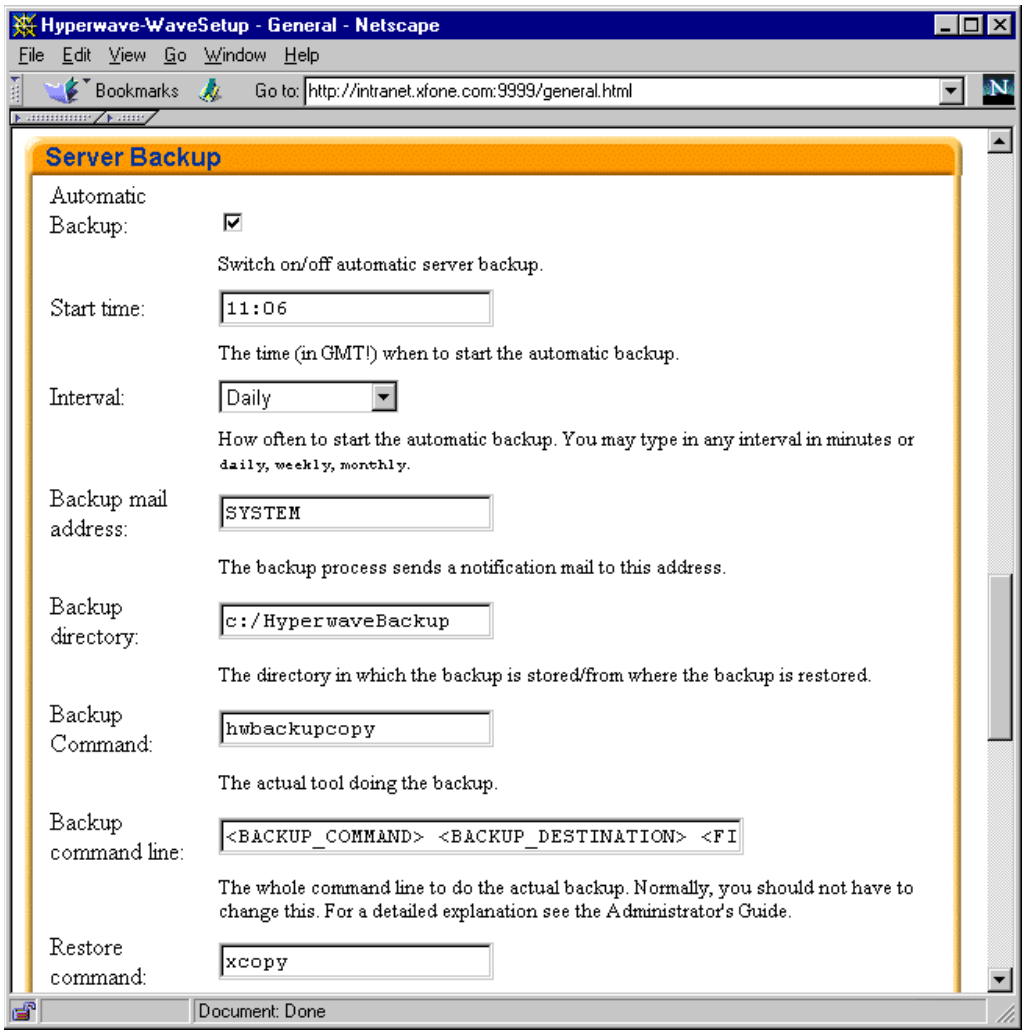


Figure 7: Configuring NT backup

3.1.1 MAKING A BACKUP OF YOUR SERVER (NT)

The backup function for the Windows NT Hyperwave Information Server can be configured using WaveSetup (see Figure 7) and lets you make automatic periodic backups of your server.

USING THE DEFAULT CONFIGURATION

Although backup for the NT Hyperwave Information Server is configurable, the default configuration that comes with the server is also usable "as is". However, by default, automatic backup is turned off. To turn it on, do the following:

1. Access WaveSetup using the URL `http://`*<my_server>*`:9999`, substituting the name of your server.
2. On the **General** page, under **Server Backup**, click on the "Automatic Backup" checkbox.
3. Click on WaveSetup's "Update general settings" button.
4. Stop and start your server to make the changes take effect.

By default, backup will be done daily at the time you restarted the server. If you want to change this schedule, use the **Start time** and **Interval** options explained below.

HOW BACKUP WORKS

To be able to configure backup, it is necessary to understand a little bit about how the backup mechanism works.

HWBackup is the main executable used to start the backup process. Its job is to make sure that the backup stores the contents of the server in a consistent state. Because backup may take hours, and during that time many changes to the data base may take place, the state at the time the backup is started is what is saved. This means that if a restore is done, any changes that took place since the start of the last backup are lost.

HWBackup's second task is to start a child process which performs the actual backup. By default, HWBackupCopy, which is supplied with the server, is called. This program is supplied because a basic NT installation does not contain programs such as TAR or ZIP to manage the task of copying several server files and directories. However, this backup command is configurable, and you can substitute the name of any program you like, as long as it can copy files.

Keep in mind that if you use a different backup program, you will have to configure the command line according to the syntax used by the command. The syntax of the command line is configured by way of placeholders which stand for the backup command, the backup destination, and the files to backup. These placeholders are, with the exception of the files to backup, also configurable.

CONFIGURING BACKUP

To configure backup, do the following:

1. Access WaveSetup using the URL `http://`*<my_server>*`:9999`, substituting the name of your server.
2. On the **General** page, under **Server Backup**, you will find the configurable aspects of backup. See the explanations below.
3. Click on WaveSetup's "Update general settings" button.
4. Stop and start your server to make the changes take effect.

CONFIGURABLE ASPECTS OF BACKUP

There are several parameters which you can configure for the backup process.

- **Automatic backup**: This checkbox allows you to turn automatic backup on and off. See page 53 for special considerations that must be made when using automatic backup.
- **Start time**: Enter the time you want backup to be started in the format YY/MM/DD HH:MM:SS. If the date part is missing, it is assumed that the current date is meant. If the time is missing, the time of the last server start is used. As an example for **Start time**, if you want to start backup every Sunday at 1:00 beginning on April 1, 1998, enter "1998/04/01 01:00".

- **Interval**: The frequency of backup is selected from the listbox. It can be done daily, each workday, weekly, or monthly.
- **Backup mail address**: The backup process sends a notification mail to this address. Note that even if the mail says that backup was successful, you should check the log files of the backup command program to see if any files could not be copied.
- **Backup directory**: The directory where the backup is stored. If the directory entered here does not exist, it is created automatically.
- **Backup command**: This is "hwbackupcopy" by default. Here you must enter a command that can copy files, e.g. "pkzip". Make sure that the command you configure is able to read open files, otherwise some parts of the server will not be backed up and the backup will not be usable. Also note that either this program must be able to create the destination directory if this directory is in the file system (e.g., not a tape destination name), or, if it cannot, you must create this directory yourself. Further, this program must return a zero on success and a value other than zero is there is an error in order for the notification mail to work.
- **Backup command line**: The backup command line consists of placeholders which are substituted with the values entered in the **Backup command** and **Backup directory** fields. `<BACKUP_COMMAND>` represents the value of the **Backup command** field and `<BACKUP_DESTINATION>` represents the value of the **Backup directory** field. `<FILES_TO_BACKUP>` cannot be configured by the user. The default value of the backup command line is `<BACKUP_COMMAND> <BACKUP_DESTINATION> <FILES_TO_BACKUP>`. If you are using a program with different command line syntax, or if you want to use other command line options, this line must be adjusted accordingly.
- **Restore command**: The command for the restore process, e.g. "pkunzip" if you used "pkzip" as backup command.
- **Restore command line**: The restore command line consists of placeholders which are substituted with the values entered in the **Restore command** and **Backup directory** fields. `<RESTORE_COMMAND>` represents the value of the **Restore command** field and `<RESTORE_SOURCE>` takes the value of the **Backup directory** field. The default value of the restore command line is `<RESTORE_COMMAND> <RESTORE_SOURCE> $HOME`. If you are using a program with different command line syntax, or if you want to other command line options, this line must be adjusted accordingly.

Note: To make file names containing spaces work, all command lines containing the variable `$HOME` *must use* `"$HOME"`.

Warning: If you have problems when using HWBackup, check that the `$HOME` ***variable is set properly, i.e. to the home directory of your Hyperwave Information Server.***

EXAMPLE CONFIGURATION

You may want to configure backup to use a program other than HWBackupCopy to copy the files. You may, for example, want to use a standard compression program to reduce hard disk consumption. Here you see an example configuration using the compression program "zip" as backup command.

Backup directory: `d:\backup`

Backup command: `zip`

Backup command line: `<BACKUP_COMMAND> -r <BACKUP_DESTINATION>\mybackup.zip <FILES_TO_BACKUP>`

Restore command: `unzip`

Restore command line: `<RESTORE_COMMAND> <RESTORE_SOURCE>\mybackup.zip $HOME`

AUTOMATIC BACKUP

When starting HWBackup automatically it is important to make sure that the process can log in to the server as a system user or else it will not be able to initiate backup.

When HWBackup is started automatically by the server's watchdog program `hwservercontrol`, it tries to autoidentify on the server as the user under which `hwservercontrol` is running. Thus autoidentification only succeeds if this user also exists on the Hyperwave server and has system rights.

Normally, `hwservercontrol` is started as an NT service running under the "System Account". Since this user does not exist on the Hyperwave server autoidenfication will fail. However, you can solve this problem as follows:

- Start the Hyperwave Information Server service as "hwsystem" (or whatever your Hyperwave system user is called).

or

- Create an arbitrary user on the Hyperwave Information Server and give that user system rights. Start the Hyperwave service as that user.

To start the Hyperwave Information Server service as different user, start the NT service control manager (**Start→Settings→Control Panel→Services**), select the "Hyperwave Information Server" service from the list and click on the "Startup" button. In that window choose "This account" in the "Log On As" section and enter the user's name and password.

HWBACKUP OPTIONS

HWBackup can also be activated in the command line to do backup or restore. The tool has several command line options which are explained below.

`-help`

Displays the list of options with brief explanations

`-version`

Shows the version string for HWBackup

`-hwhost` *name*

The name of the server to backup

`-hwport` *port*

The port number to connect to

`-backup`

Puts the tool into backup mode

`-restore`

Puts the tool into restore mode

`-identify` [`'`*user*[*password*]`'`]

You must specify your user name/password combination with this option

`-mailto`

Email account to send notification mails to

`-command` *command*

Command executed for performing backup task, e.g. "zip"

`-logfile` *file*

Path and name of the log file

`-targetdir` *targetdir*

The directory where the backup is stored (used with backup only)

`-sourcedir` *sourcedir*

The restore source directory (used with restore only)

`-verbose`

Enables verbose output

USING THE COMMAND LINE FOR BACKUP

Although you can have backup run automatically, it is also possible to start it with the command line. To do this, enter

```
hwbackup -backup
```

in the command line.

The `-backup` option causes the tool to use the command line configured with the option **Backup command line** in WaveSetup. All settings made in WaveSetup are written into Hyperwave's main configuration file, `.db.contr.rc`. Note that you can override the configured command line by entering your preferred options in the command line.

USING THE COMMAND LINE FOR RESTORE

Restoring the server can only be done using the command line. It is done as follows:

1. Stop the server.
5. In the command line, enter

```
hwbackup -restore
```

The actual command line for this operation is taken from the settings made using WaveSetup (**Restore command line**), which are explained above. These settings are written into Hyperwave's main configuration file, `.db.contr.rc`. Because it is possible in the case of a restore that all of your server's files have been lost, it may not be possible to read this information out of the configuration file. If this is the case, you can enter the required command line directly, which overrides the contents of `.db.contr.rc`. An example for the command line in this case is:

```
hwbackup -restore -sourcedir d:\backup -command xcopy
```

See above for an explanation of the command line options for HWBackup.

Warning: The restore process overwrites everything that has been changed on the server since the last backup!

3.1.2 MAKING A BACKUP OF YOUR SERVER (UNIX)

The UNIX Hyperwave backup program is called `hgbackup` and uses `tar` to archive the contents of the server. Besides calling `tar`, it performs some additional actions that guarantee the consistency of the backup. The server can be active during the backup operation when using this script.

The restore program is called `hgrestore` and also uses `tar` to restore the files from tape. During the restore process the server is stopped and after reorganizing the database contents it is started again.

If you only want to back up certain collections and not the entire server, or if users want to back up collections they are working on, the programs `hifexport` (for backing up) and `hifimport` (for restoring) are very useful. These tools are described on page 104.

USING HGBACKUP AND HGRESTORE

`hgbackup` and `hgrestore` are command line tools with the usage

```
hwsystem/bin/scripts/hgbackup [-h|-help] [-test]
```

`-h`, `-help`	prints this message
`-test`	reads configuration file `.hg-backup.rc` and prints variable settings

Both the `hgbackup` and `hgrestore` scripts can be configured by editing a file named `.hg-backup.rc` located in `hwsystem`'s home directory. If you have no such file, copy the sample file `hg-backup.rc` located in the directory `$HOME/samples/hg-backup.rc` to

the file $HOME/.hg-backup.rc. Lines in the file starting with '#' are comments and are ignored. All other lines are of the form *variable = value*.

Below is a list of the variables which can be configured including a short explanation of each.

GENERAL VARIABLES USED BY HGBACKUP AND HGRESTORE

backup_log: name of the log file for the output of HG_backup_command_line. The default is HG-backup.log in your home directory.

Example: backup_log = HG-backup.log

restore_log: name of the log file for the output of HG_restore_command_line. The default is HG-restore.log in your home directory.

Example: restore_log = HG-restore.log

mail_to: address to mail error and success messages of the hgbackup script to (optional).

Example: mail_to = hgmgr@iicm.tu-graz.ac.at

backup_dev: special device file for the backup medium. Default is dev/tape. You can also enter the full path name of your tar file, e.g. backup_dev=/usr/hwsystem/test.tar

backup_command: is your default backup command. Default is /bin/tar.

Note: If you prefer another backup method (e.g. cpio) you will have to change the variables HG_backup_command_line *and* HG_restore_command_line *(see below).*

Variables for remote backup/restore:

backup_host: remote host containing the backup device.

rsh_command: remote shell command. Default is bin/rsh.

Note: *On System V operating systems this command is (sometimes) named* remsh. *On some machines* rsh *is the "restricted shell". Do not use it!*

backup_user: user on the remote host (backup_host).

Note: *The* .rhosts *file in the home directory of* backup_user *must contain an entry for* local-host *and* local-user (hwsystem)*, because* hwsystem *has to execute a remote shell on* backup_host*. If* backup_user *is omitted,* hwsystem *must have a valid account on the backup host.*

Example: backup_user = hwsystem

dd_command: command used on backup host to write the data to the backup device. Default is /bin/dd.

Note: *Parameters for the* ***dd_command*** *must be inserted in the* HG_backup_command_line *and* HG_restore_command_line *variables below.*

Backup and restore command lines

You can use the above variables (preceded by a '$') when specifying these lines. They will be replaced by their values. '<FILES_TO_BACKUP>' will be replaced by the files to backup, so don't remove it from the command line.

- For a local backup use:

```
HG_backup_command_line = $backup_command cvBbf 20 $backup_dev
<FILES_TO_BACKUP>
```

- For a remote backup use:

```
HG_backup_command_line = $backup_command cvBbf 20 -
<FILES_TO_BACKUP> |
$rsh_command $backup_host -l $backup_user $dd_command bs=20b
of=$backup_dev
```

- for local restore use:

```
HG_restore_command_line = $backup_command xvf $backup_dev
```

· for remote restore use:

```
HG_restore_command_line   =   $rsh_command   $backup_host   -l
$backup_user     $dd_command     bs=20b     if=$backup_dev     |
$backup_command xvf –
```

3.1.3 BACKING UP THE ORACLE DATABASE

If you are using Oracle instead of Hyperwave's native database, you cannot use the Hyperwave backup tools to make a backup of your server. You must use the tools provided by Oracle in order to perform data backup tasks for the underlying ORACLE database.

There are two different ways of backing up the ORACLE database depending on the server availability:

OFFLINE BACKUP

If there is a time each day (e.g. at night), where there is no need for the server to be available, it is possible to do the backup when the instance is shut down, so no changes to its files can occur. Thus backup simply copies all important files to a tape-device or similar. The disadvantage of this appoach is the down-time of the server which increases with the amount of written data.

Offline backup can be performed in two different ways:

· by Database Export (Logical Backup)

This backup mode also allows incremental or cumulative export, reducing the amount of data written at one time.

· by File System - (Physical-) Backup

This is the simplest approach. For databases where most tables are changed in the between-backups-period, it isn't of any disadvantage compared to incremental or cumulative backup, because each table with at least one changed row is exported.

ONLINE BACKUP (ARCHIVELOG BACKUP)

If your application does not allow an offline backup for availability reasons, the second possibility is to do an ARCHIVELOG backup. During ARCHIVELOG backups, there is no need to shut down the database. In this mode, the redo information written by the database LOGWRITER is saved in addition to the data files and control files. That way the information changes applied to the data files during the backup process are stored in the archived REDO LOG files and for this reason, they can, in case of a disaster, be reconstructed.

Note: For the ORACLE Point-In-Time Recovery feature the database in must be run in ARCHIVELOG mode.

An explanation in more detail for each backup mode can be found in the Oracle8 DBA Handbook in Chapter 10: Optimal Backup and Recovery Procedure. Also see the manual Oracle8 Backup and Recovery, Release 8.0.

3.2 SETTING UP USER ACCOUNTS AND GROUPS

One of Hyperwave's most useful features is that it allows you to give users accounts on your server and to assign access rights to documents and collections. This powerful feature makes it possible to restrict users so they can only edit selected parts of the total information space on the server.

Hyperwave Information Servers organize their content into a collection hierarchy. This relatively strict structuring is what allows Hyperwave's system of access rights to work so well. If, for example, you are running a server for a large company with several departments, you can give each department its own collection, give each employee a Hyperwave account, create a group for each department and put the employees into the corresponding group. You can then specify the access rights for each collection such that only members of a department have write access to the documents belonging to that department. Of course it is possible to continue this process by dividing the information for a department into documents pertaining to different projects and then defining groups and access rights accordingly.

Note that only the Hyperwave system administrator and members of the group "system" can insert, edit and delete user accounts and groups.

You can use WaveMaster, Hyperwave's WWW gateway, to set up users and groups and edit their attributes. It is also possible to use the command line tool `hwadmin` for the same purpose. Both are described below.

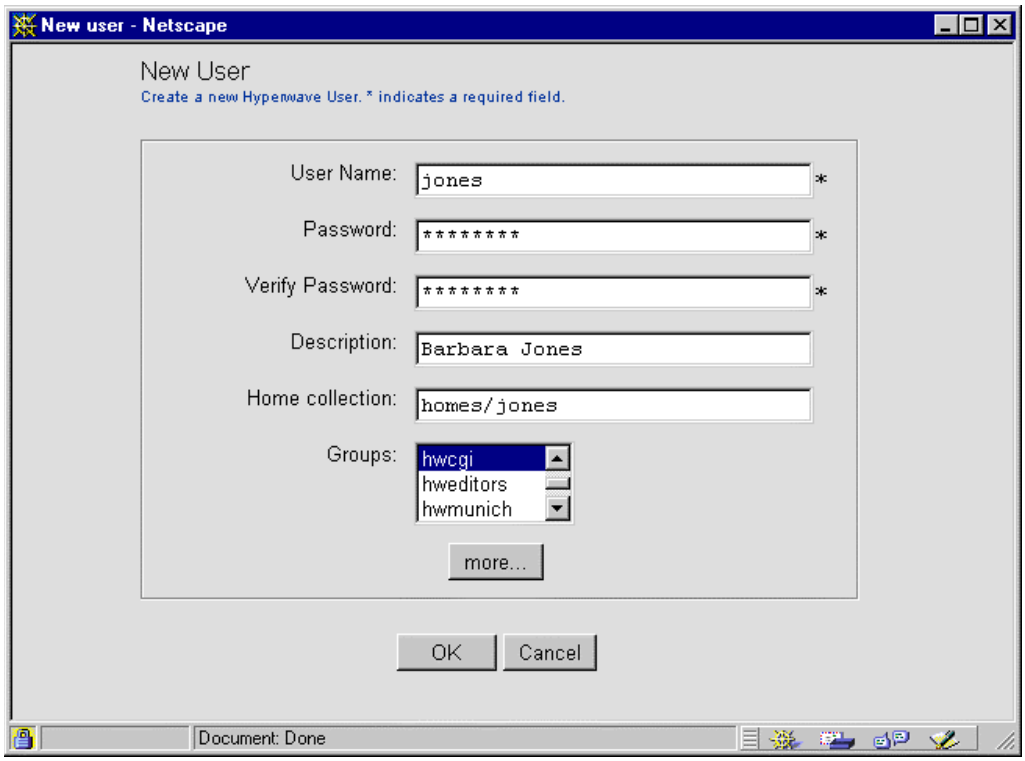


Figure 8: Inserting a new user

3.2.1 USER ADMINISTRATION WITH WAVEMASTER

The WaveMaster, Hyperwave's WWW gateway, has built-in features for administrating users and groups, which means that you can use any Web client for this purpose (see Figure 8). To do this, access your server with a Web client and identify as a system user (member of the group "system", which means you are allowed to read and edit all objects on the server, including user accounts). If you just started your server for the first time, there exists only one Hyperwave account, for which the user name and password vary according to platform as follows:

INITIAL USER NAME AND PASSWORD FOR UNIX

For the UNIX platforms the password depends on whether the UNIX system uses shadow passwords or not.

2. In most cases, the user name and password are taken from the UNIX account under which Hyperwave Information Server was installed.
3. In the case of UNIX systems with shadow passwords, the password field in `/etc/passwd` cannot be read, and thus the password of the UNIX account cannot be used. In this case, the user name is taken from the account and the password is "hwsystem". For security reasons it is recommended to change the password as soon as possible using **Site→Change Password**.

INITIAL USER NAME AND PASSWORD FOR NT

For Windows NT, both the user name and the password are "hwsystem".

This initial user belongs to the group "system". After you have identified, click on the **Admin** icon to display a group of index cards, some of which allow you to access information about the server, and some of which let you administrate users and groups. How to add, delete and edit users and groups is explained below.

3.2.2 GROUPS

You can use Hyperwave to maintain user groups in order to grant or deny access rights to certain parts of the information space. Users may be made members of these groups, which are organized into a hierarchy of groups and subgroups. A group may have multiple parent groups and multiple subgroups, but no group may belong, either directly or indirectly, to itself. A group inherits the rights of the group(s) of which it is a member.

3.2.2.1 ADDING, EDITING AND DELETING GROUPS

Hyperwave Information Server allows system users to add, edit and delete groups.

LISTING GROUPS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Groups→List Groups**. A window containing a list of all the groups on the server appears. Click on "OK" to close the window.

ADDING GROUPS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Groups→New Group**. The "New Group" window appears. It is required that you enter a name for the group. You can optionally enter a description. If you enter a parent group, the group you are creating inherits the rights of that group. You can select one or more users from the listbox to be members of the group. See below for an explanation of group attributes.
4. Click on the "OK" button to create the new group.

EDITING GROUP ATTRIBUTES

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Groups→List Groups**. A window containing a list of all the groups on the server appears.
4. Click on the "Edit" button next to the name of the group you want to edit. A window containing the attributes of the group appears.
5. Change or add the desired attributes. Click on the "More" button to add further attributes. See below for an explanation of all group attributes.
6. Click on the "OK" button to apply your changes.

DELETING GROUPS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Groups→List Groups**. A window containing a list of all the groups on the server appears.
4. Use the checkboxes to select the groups you want to delete.
5. Click on the "Delete" button. The selected groups will be permanently deleted.

SHOWING GROUP MEMBERS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Groups→List Groups**. A window containing a list of all the groups on the server appears.
4. Click on the "Edit" button next to the name of the group whose members you want to view. A window containing the attributes of the group appears.
5. Click on the "Group Members" button to see a list of the users and groups contained in this group.

GROUP ATTRIBUTES

The following is a list of attributes of group objects:

· **Description**

A short description of the group.

· **Group**

The value of this attribute must be the name of an existing group. The group you are editing becomes a subgroup of this group and inherits its access rights.

· **Owner**

The owner of the group.

3.2.3 USERS

3.2.3.1 ADDING, EDITING AND DELETING USERS

Hyperwave Information Server allows system users to add, edit and delete user accounts.

LISTING USERS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Users→List Users**. A window containing a list of all the user accounts on the server appears. Click on "OK" to close the window.

ADDING USERS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Users→New User**. The "New User" window appears (see Figure 8). It is required that you enter a name for the user, a password, and that you verify the password. You can optionally enter a description or a home collection, or select one or more groups from the listbox.
4. Click on the "more" button to add further attributes or to create a home collection for the user. See below for an explanation of all user attributes.
5. Click on the "OK" button to create the new user.

EDITING USER ATTRIBUTES

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Users→List Users**. A window containing a list of all the user accounts on the server appears.
4. Click on the "Edit" button next to the name of the account you want to edit. A window containing the attributes of the user account appears.
5. Change or add the desired attributes. Click on the "More" button to add further attributes. See below for an explanation of all user attributes.
6. Click on the "OK" button to apply your changes.

DELETING USERS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Users→List Users**. A window containing a list of all the user accounts on the server appears.
4. Use the checkboxes to select the users you want to delete.
5. Click on the "Delete" button. The selected users will be permanently deleted.

ASSIGNING USERS TO GROUPS

1. Log in to the server as a "system" user.
2. Click on the item **Authoring** on the toolbar.
3. Select **Admin→Assign Users and Groups**. A window appears where you can either assign users to or remove users from groups.
4. To assign users to groups, select one or more users and one or more groups in the top part of the window and click on the "OK" button. To remove users from groups, select one or more users and one or more groups in the bottom part of the window and click on the "OK" button.

USER ATTRIBUTES

The following is a list of attributes of user accounts:

- **Encrypted password**

  The encrypted password is stored under UNIX in the file `/etc/passwd` or you can get the encrypted password with the SUN Yellow Pages command `ypcat passwd`. The advantage of this method is that the system administrator can set up a user with the user's UNIX password without knowing the password itself.

- **Host**

  These are the hosts from which the user is automatically identified. They can be entered using the domain name or IP address in the form *user@host*. Automatic identification only works if the password of the user is the same as the password of the underlying account on the host machine.

- **Group**

  Here you can enter the name of an existing group which you want to make the user a member of.

- **Account**

  In Hyperwave, documents can be given a price and users can have a certain amount of virtual money to access such documents. The **Account** attribute is where the virtual money is given to the user. Only members of group "system" are able to change this attribute. The server decrements this value whenever the user accesses a priced document until the value is zero. If the value is already zero, the document is not transferred and the server sends an error code. This number must be entered in hexadecimal. Its value is `0x00000000` by default.

- **Owner**

  The owner of the user record.

- **Description**

  A short description of the user, usually the user's full name.

- **Password**

  The user's password in plain text (it is not echoed when it is entered).

- **Home**

  Home is a collection which is designated as personal space for users where they can insert their own documents or make references to documents on other servers. This entry is optional. The collection can be inserted into the database (using `hwinscoll` or WaveMaster) before the user's account is created or can be inserted at the same time the new user account is inserted. What must be entered here is the unique name of the collection, which you can look up by viewing its attributes. This can be done by e.g. clicking on the attributes link when viewing the collection in the WaveMaster.

- **Language**

  This attribute is used to set the user's preferred language for the WaveMaster interface. The value must be one of the Hyperwave language prefixes (see page 89).

- **PrefMimeTypes**

  The value of this attribute is used to determine which document is retrieved when the user accesses an alternative cluster (see the *Hyperwave User's Guide*).

  The syntax for the entry to be made in this field is as follows:

  ```
  MimeType[";Quality=number"]*(,MimeType[";Quality=number"])
  ```

  For example:

  - `text, image, application/postscript`

    This means that the user prefers documents of type text to images, and images to PostScript, and PostScript to any other type.

  - `image;Quality=50`

    This means that images are preferred to any other type and if more than one image is available the one with the quality nearest to 50 is chosen.

Note: *Some fields (e.g.* Password, Group, Description *and* Host*) can exist multiple times for the same user.*

3.2.4 HWADMIN

`hwadmin` is a menu-driven command line tool which is used to administrate users and groups. See page 57 to find out about how users and groups work in Hyperwave and what attributes they can have.

3.3 SETTING ACCESS RIGHTS IN HYPERWAVE

This section explains how to set access rights for Hyperwave objects.

THE RIGHTS WIZARD

Though the syntax used in the **Rights** attribute is explained here in great detail, it not absolutely necessary to memorize the exact syntax when giving this attribute to a Hyperwave object using a web browser. This is because the WaveMaster offers you the Rights Wizard, a tool which lets you select the users and groups you want to give rights to from a list of all the existing users and groups, thus greatly simplifying specification of this attribute.

ACCESS RIGHTS SYNTAX

Access rights are specified in the **Rights** attribute of Hyperwave objects (documents, collections and anchors). The attribute value is composed of read (`R:`), write (`W:`) and unlink (`U:`) permission fields, separated by semicolons. For each field, the value "`a`" means that the author of the object has access, '`u` *users*' means that the specified users (user names separated by blanks) have access, and '`g` *groups*' means that the members of the specified user groups (group names separated by blanks) have access. Each field may appear a maximum of one time, and within the field, field values may be separated by commas, e.g., '`R:a,u` *users*`,g` *groups*'.

The **Rights** attribute can only be modified by the author of a given object or members of user group "system".

By default (that is, when no **Rights** attribute is present or a field is empty or missing) only the author of the object and members of the group "system" have write and unlink permissions while all users (including anonymous users) have read permission. In principle, the **Rights** attribute allows you to reduce the set of users who have read access, to enlarge the set of users who have write access, and to reduce the write access set to a set of users with unlink access.

Write access implies read access. Unlink permissions are only meaningful for collections, as they control the right to unlink (remove) something from that collection. By default (that is, if the unlink field is not present) users with write permission have unlink permission.

If the write field is present and the unlink field is not, the users specified by the write field also have unlink permission and can thus delete the object from the collection.

In addition to the **Rights** attribute, access is controlled by the **TimeOpen** and **TimeExpire** attributes (see the *Hyperwave User's Guide* for a description of these attributes). After the date specified by **TimeExpire** the object becomes invisible to users who do not have write access to the object (the object is, however, not physically removed from the database). Similarly, an object is invisible before the date specified by **TimeOpen** to users without write access to the object.

Summarizing, write access is granted if:

1. the user has system privileges (that is, is a member of user group "system"), or
2. the user is the author (owner) of the object, or
3. the user is a member of the users or user groups specified in the write field (`W:`) of the object's **Rights** attribute.

Read access is granted, if:

1. the user has write access (see above), or
2. the time constraints are met (**TimeOpen**, **TimeExpire**), and
    1. The read field (`R:`) of the object's **Rights** attribute is not present or empty (that is, unlimited read access is granted), or
    2. the user is a member of the users or user groups specified in the read field of the object's **Rights** attribute.

Unlink access is granted if:

1. the user has system privileges (that is, is a member of user group "system"), or

2. the user is the author (owner) of the collection to unlink from, or
3. the user has write access to the collection to unlink from (see above), and
   1. the unlink field (`U:`) of this collection is not present or empty, or
   2. the user is a member of the users or user groups specified in this field.

   As a special case, "`U:a`" refers to the author of the object to be deleted (not the author of the collection), so authors are allowed to delete their own objects only, even if they have write access to the collection.

Write (unlink) access for a document implies write (unlink) access for all anchors attached (that is, even for private anchors of other users). The access permissions contained in the **Rights** attributes are checked when the object (document, collection, anchor) is requested. In other words, if you do not have read permission for a document, you will not even be able to access its meta-data, let alone the document itself.

INHERITING RIGHTS

Two more permission fields are allowed in the **Rights** attribute, although they do not directly control access to the object record. The `I:` field can be used to control the inheritance of a collection's access rights to its children. It is meaningful only for collections. It is not interpreted by the server but by the WaveMaster, which uses it to make the appropriate suggestion as default Rights attribute. The `I:` is followed by a combination of the letters `R`, `W`, `U`, `L` and `I`. For example, `I:R` means that only the `R:` portion of the collection's Rights attribute should be copied to the client's **Rights** attribute. By default (that is, when the `I:` field is missing), clients should copy (inherit) all fields (equivalent to `I:RWULI`).

THE LICENSE FIELD

The `L:` field is meaningful for documents only, and is used to limit simultaneous access to resources. The L stands for "licenses" and the idea is that system administrators can buy licenses for certain collections of information (such as electronic books, journals, newspapers, databases) from the owner of the information (for example, a publisher), similar to how a library would buy a book or a CD-ROM. The idea is that if n copies of the book (licenses of the collection) have been bought, the book (collection) can only be read by n people simultaneously.

Hyperwave supports this model with the **License** attribute of a collection and the `L:` field of the **Rights** attribute of individual documents of that collection. When a user retrieves a document from the licensed collection for the first time, one license is assigned to that user, and thus locked for others. The user may continue reading other documents from that collection. If no retrieval of documents by that user happens for a certain timeout period the license is unlocked (analogous to a book being returned to the shelf). For example, the **License** attribute of Meyer's 10-volume Encyclopedia on IICM's Hyperwave Information Server looks like this:

```
License = 0x0000012c 0x00000003 0x00000014
```

The first of the three hexadecimal numbers specifies the timeout period- in our example 5 minutes (hexadecimal 12c is 300 seconds). This represents the length of time that a user can lock the collection without retrieving documents. A timeout value too large blocks the collection unnecessarily, while a timeout value too small effectively circumvents the license restrictions. The second hexadecimal number specifies the number of licenses or simultaneous accesses to the collection - in our example, 3. The fourth user would be denied access.

The last hexadecimal number specifies the number of documents a user is allowed to retrieve in a single "grab" of the license. This is a measure against somebody trying to download the whole collection (of possibly copyrighted material) and blocking it for a long time for others. In our example, a user may look at 20 encyclopedia entries before having to return it. Thus, the maximum time a single user can keep the encyclopedia is $5 \times 20 = 100$ minutes. Also, the time to download the whole encyclopedia (43,820 documents) is significantly increased (left as an exercise for the reader!).

Every document in the encyclopedia must contain a reference to the collection with the **License** attribute in the "`L:`" field of its **Rights** attribute. In our example, the name of the collection is "ref.m10", so the **Rights** field would have to contain the following value:

```
L:ref.m10
```

so that every access to a document in the collection is taken into account, regardless of how it was found (including searching and link following). It is not necessary to directly access the collection with the **License** attribute to enable the license mechanism.

3.3.1 EXAMPLE VALUES FOR THE RIGHTS ATTRIBUTE

· `R:g abcd,u xyz`

Members of user group `abcd` and user `xyz` have read permission, only the author of the object has write and unlink permissions (the write field is not present and thus the defaults are used).

· `R:g a1 a2;W:g a1`

User groups `a1` and `a2` have read permission, but only group `a1` has write and unlink permissions.

· `W:g b1 b2;U:g b1`

All users have read permission (default; read field is empty), and groups `b1` and `b2` are granted write permission, but only the members of `b1` may also remove objects from this collection (the unlink field is ignored for objects which are not collections).

· `R:a`

Only the author has read (and by default write and unlink) permission.

· `W:a`

This is the default setting that is used when the Rights attribute is not present: everybody can read, but only the author may write and unlink.

· `I:R;R:g abc xyz;W:g xyz;U:a`

This setting of a collection is useful in situations like electronic discussions, where only members of groups `abc` and `xyz` are allowed to read the contributions contained in the collection and only members of `xyz` are allowed to write contributions, but no user should be allowed to unlink the contributions of others (`U:a` means that authors can unlink only their own documents from this collection). `I:R` prevents unwanted copying of the `W:` field to documents inserted into this collection, so that write access for documents in the collection remains with the authors only.

Attributes (including the **Rights** attribute) may be changed and added interactively using any web client. If you have to change a number of objects in a similar fashion, you may use the `hwmodify` command, either by itself or in a script.

3.4 USER MAPPING

Hyperwave can be configured so that users who connect to the server using a certain machine or a machine in a specific domain are automatically identified under a specified user name.

To use this feature you have to configure it using WaveSetup or the configuration file `.db.contr.rc` (found in `~hwsystem`'s home directory), and create a file which contains a user-host table. See below for user-host table syntax.

USING WAVESETUP

To access WaveSetup, Hyperwave's configuration tool, see page 8. Click on the "Network" tab. Under **User Mapping** you can enter the name of your user-host table file and edit the file (see Figure 9).

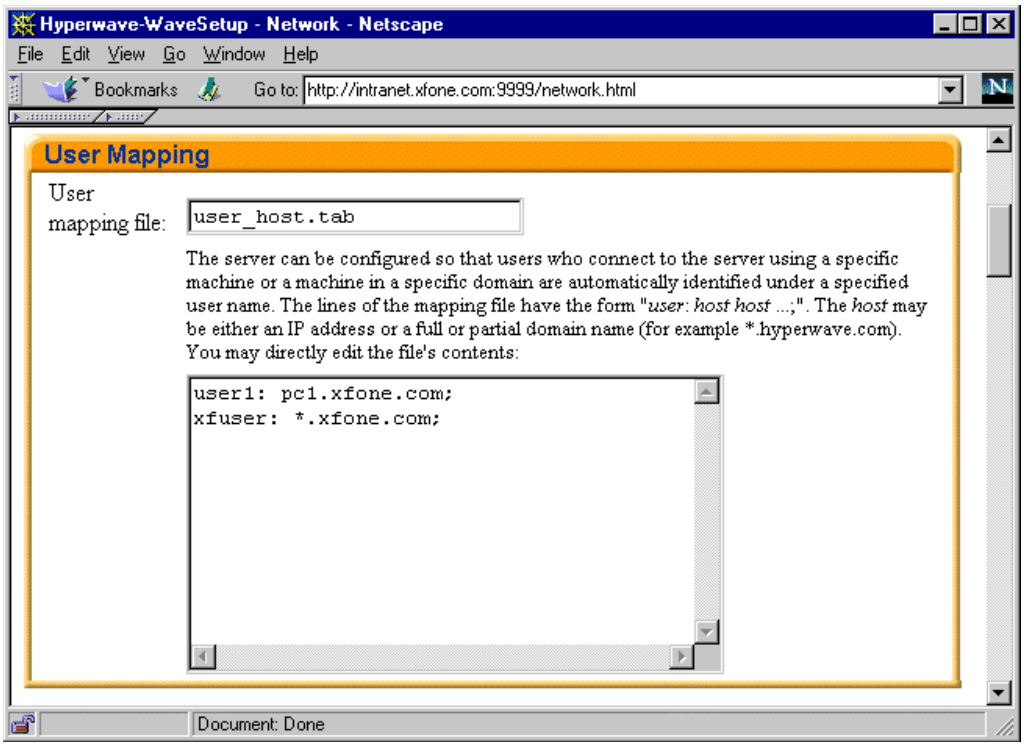


Figure 9: Configuring user mapping with WaveSetup

USING .DB.CONTR.RC

To configure user mapping in `.db.contr.rc`, enter the line

`<scope>::USERMAPPING=`*file*

where *file* is the name of your user-host table file, specified relative to `~hwsystem/dbserver` or as an absolute path. `<scope>` is the name of your WaveMaster process, usually "`WAVEMASTER`".

USER-HOST TABLE SYNTAX

The user-host table file consists of one or more lines, each of which maps at least one domain to a user name (this user name must exist in the Hyperwave user database). Each line starts with a single user name followed by a colon. This is followed by a list of domain names or IP addresses separated by spaces. Each line ends with a semicolon.

An asterisk ("`*`") can be used at the beginning of a domain name to map all the users at machines in a particular domain. For example, the line

```
testuser: pc1.mydomain.com;
```

identifies users using one specific computer as `testuser`, but

```
testuser: *.mydomain.com;
```

identifies people at all the machines in the domain `.mydomain.com` as `testuser` when they access the server.

Assignment of users is line by line top-down and from left to right in each line. The first entry found determines the user, thus you can also refuse some machines. Here are some examples:

```
anonymous: refused.machine.domain;
user1:     *.domain 150.20.30.40;
```

3.5 EXTERNAL IDENTIFICATION

UNDER WINDOWS NT

The Hyperwave NT server comes with an external identification gateway which is installed automatically when you install or upgrade the server. This gateway allows all users who have an account under NT to log in to the server using their NT user names and passwords.

External identification is turned off by default. You can switch it on or off using WaveSetup as follows:

1. Access WaveSetup with a browser with the URL `http://`<your_server_name>`:9999`
2. Select WaveSetup's **General** tab.
3. Click on the checkbox under **External Authentication**.
4. Click on the **Update General Settings** button to make the changes in Hyperwave's configuration file.
5. **Stop and restart your server** by clicking on the **Restart server now** button in WaveSetup. This must be done for the changes to take effect.

***Note**: If your group or user names under Windows NT contain umlauts or other special characters, they are converted using SGML to ISO syntax (e.g. "ä" is mapped to "ae").*

***Note**: If you are using the Hyperwave NT identification gateway, make sure that you have disabled the "guest" account on the NT server (this account exists automatically when Windows NT is installed). Otherwise all unknown users will be mapped to "guest" by the NT subsystem and they will be valid users for the Hyperwave Information Server.*

ORDER OF DATABASES HYPERWAVE SEARCHES

When external identification is being used with Hyperwave for Windows NT, the order of the databases Hyperwave looks through when identifying a user is as follows:

1. The local Hyperwave user database.
2. The local Windows NT user database on the host Hyperwave is running on.
3. The NT domain user database of the domain the Hyperwave host is in.
4. User databases in NT friend domains.

Group information for NT groups is also sent to the server, e.g. if an NT user belongs to the group "system", he or she will also be logged in as a system user on the server.

Note that external identification may, in some cases, take some time.

NT ID INFORMATION FROM OTHER DOMAINS

It is possible to configure identification gateways on NT machines other than the one the server is running on in order to get identification information from other domains. Not only user but also group membership information is transmitted over these gateways. This is done as follows:

Configure a separate NT ID gateway on the primary domain controller of each domain you want to query by typing

```
hwidgate -install
```

at the command prompt and adding the line

`HGSERVER::AUTHENTICATION[`*domainname*`] = TCP` *domaincontroller* `3000`

to your Hyperwave Information Server configuration file (`.db.contr.rc`) and changing *domainname* and *domaincontroller* appropriately.

***Note:** It is also possible to configure an NT identification gateway for use with a server running on a UNIX platform.*

A second possibility for getting user and group membership information from another domain is to configure the other domain as trusted and trusting and make sure that the user account under

which the Hyperwave server (and thus the NT ID gateway) is running has the appropriate access rights to read group memberships in the other domain.

UNDER UNIX If you have a database of users (yellow pages, X.500, etc.) it is possible in Hyperwave to utilize this database to identify users rather than to set up accounts for all of them. Hyperwave is equipped with an interface for this type of gateway and comes with a yellow pages gateway. For information on how to program and configure a gateway for external identification, see the *Hyperwave Programmer's Guide*.

3.6 ADDING HYPERWAVE TO AN EXISTING WEB SERVER (WAVEBACK)

WaveBack makes it possible to run a standard Web server, listening on port 80 (HTTP), and transparently access documents on a Hyperwave Information Server. This means that with a few simple steps, Hyperwave can be added to your existing Web server infrastructure without changing it. Documents may reside on either server. You may or may not choose to gradually move them to the Hyperwave side to take advantage of the powerful document management features of Hyperwave. Add-ons and plug-ins into your existing server remain functional.

3.6.1 INSTALLING CGI WAVEBACK

UNIX The server-to-server interface is set up by copying the binary `waveback` and its configuration file `.waveback.rc` to your WWW server's CGI directory and editing the configuration file (see below). The binary and the configuration file are on the CD in directory `/unix/`*<your_architecture>*`/` in the zipped tar file `waveback.tgz`. How to unzip and untar the file for your architecture is explained in the README file in the same directory.

WINDOWS NT Installation of WaveBack for Windows NT is similar to installation for UNIX except the required files, `waveback.exe` and `.waveback.rc` are found in the `server/bin` directory of the installed server. Copy these two files to your WWW server's CGI directory and edit the configuration file (see below).

3.6.1.1 CONFIGURING CGI WAVEBACK

The configuration file `.waveback.rc` contains lines of the form `variable=value`. The following parameters can be configured:

`HWHost`: IP or domain name address of the hidden Hyperwave Information Server.

Example: `HWHost=www.hidden.com`

`HWPort`: Port on which the hidden server listens for HTTP requests.

Example: `HWPort=8080`

`HWScript`: URL of the WaveBack script.

Example: `HWScript=http://www.foo.com/cgi-bin/waveback`

3.6.1.2 ACCESSING HYPERWAVE DOCUMENTS THROUGH WAVEBACK

To access documents on a Hyperwave Information Server through WaveBack, the **Name** or **GOid** attribute of the object you want to access must be added to WaveBack's URL.

Example:

If `http://www.foo.com/cgi-bin/waveback` is the URL of WaveBack and you want to retrieve the document named "some/path/some/doc" from the server, use the URL `http://www.foo.com/cgi-bin/waveback/some/path/some/doc`.

Note: When using the CGI interface to access a Hyperwave Information Server through a WWW server, it is not possible to identify on the Hyperwave server for reasons having to do with the CGI specification. To access restricted information you must connect directly to Hyperwave.

Note: With WaveBack for Windows NT, the WWW server should open input and output for CGI scripts in binary mode and not text mode because otherwise problems may occur with binary data (images, etc.).

3.6.2 APACHE WAVEBACK

Using Apache's API it is possible to write your own modules to extend the server. The WaveBack implementation as Apache module provides two advantages over the CGI implementation:

- The module has full access to the HTTP request and is thereby able to pass the full header (including authentication, cookies, etc.) to Hyperwave (in other words, using the Apache module implementation there are no functional restrictions on the Hyperwave Information Server).
- Better performance since there's no need to start an external process for every request.

Note: Apache WaveBack is only available for UNIX.

There are two ways to use the WaveBack Apache module:

WaveBack allows you to configure a virtual path on your Apache Server which is mapped to Hyperwave's root collection (if for instance the virtual path is configured as `http://www.foo.com/waveback`, a request to `http://www.foo.com/waveback/some_document` will fetch the document named "some_document" from Hyperwave and pass it on to the client). Links in documents fetched via this virtual path that point to documents on the same Hyperwave Information Server as the one containing the link are replaced, so that the destination documents are also fetched via this virtual path.

In addition, WaveBack allows you to configure paths which are mapped directly to the Hyperwave Information Server (if for instance "some_collection" is configured as such a path, a request for `http://www.foo.com/some_collection/some_document` will fetch the document named "some_collection/some_document" from Hyperwave. Additionally, you may configure whether links in documents fetched via such paths are replaced or not (if they are replaced they will point to the Apache server, if not they will point directly to the Hyperwave server). This allows you to move parts of the Apache document tree to the Hyperwave server and still have the old URLs valid.

3.6.2.1 INSTALLATION

To integrate the WaveBack module into the Apache server, you'll have to recompile your Apache server (which of course means you need the source code).

WaveBack was developed and tested using Apache 1.2.0. It may work differently or it may not work at all with other Apache versions.

You must compile Apache with **gcc**. The WaveBack library was compiled with **gcc**; you'll get undefined references if you use any other compiler.

You must have g++ installed (you need at least libstc++.a) since the WaveBack library contains C++ code and references streams and other C++-stuff defined in there.

1. Unpack the WaveBack tar file (`gzip -cd waveback_apache.tar.gz | tar xvf -`).
2. Copy `mod_waveback.c` and `libWaveBack.a` to the Apache source directory.
3. Edit the `Configuration` file (found in the Apache source directory):

   a. add the line

   ```
   Module wb_module mod_waveback.o
   ```

   to the Modules section. Note that the modules are listed in reverse priority order, so if you list WaveBack as first module, its behavior may be overridden by all other modules. If you list it as last module, its behavior may override that of all other modules (see the Apache documentation for details).

   b. add

   ```
   -L. -lWaveBack -lstdc++
   ```

   to the `EXTRA_LIBS`.
4. Type `Configure`.
5. Type `make`.

3.6.2.2 CONFIGURATION

WaveBack reads the following parameters from Apache's configuration file `conf/httpd.conf`:

- `HWHost` <*hostname*>: the IP or domain name address of the Hyperwave Information Server.

  Example: `HWHost www2.foo.com`
- `HWPort` <*portnumber*>: the port number of the Hyperwave Information Server's WaveMaster.

  Example: `HWPort 80`
- `HWPath` <*url*>: specifies the "virtual" path (the full URL including protocol and host) to the Hyperwave Information Server.

  ```
  HWPath http://www.foo.com/waveback
  ```
- `HWMap` <*path*> [`replace|noreplace`]: specifies an additional path which is mapped to the Hyperwave Information Server in the background. If the path is configured with the `replace` option, links to the server are replaced so that they also point to the Apache server. If the path is configured with the `noreplace` option, links in documents fetched from the Hyperwave server are not replaced (i.e. they point directly to the Hyperwave server). Note that in contrast to the virtual path configured with `HWPath`, paths specified in `HWMap` will not be stripped off the request.

  Examples: `HWMap /some_dir noreplace`

  `HWMap /some_other_dir replace`
- `HWTempDir` <*path*>: the path for temporary files. Documents fetched from the Hyperwave Information Server are stored in a temporary file before they are sent to the client. So if you want to serve larger documents (for example audio or video clips) from Hyperwave via WaveBack, be sure there is enough temporary space. The default setting is `/tmp`.

  Example: `HWTempDir /usr/tmp`

3.7 HOW TO CONNECT DATABASES TO HYPERWAVE USING NETDYNAMICS (UNIX)

3.7.1 REQUIREMENTS

To run NetDynamics you need Hyperwave version 2.5 or higher running on a system supported by NetDynamics. See `http://www.netdynamics.com`.

You have to download the appropriate NetDynamics package from `http://www.netdynamics.com/download/download.html`.

3.7.2 INSTALLING NETDYNAMICS

1. Login as a Hyperwave system user

   `$HOME>`

2. Create directory which will contain the NetDynamics installation

   e.g. `$HOME>mkdir nd311` (to install NetDynamics version 3.11)

3. Unpack the NetDynamics archive file, which is in the form `nd`<*version*>`_`<*os*>`.tar` (e.g. `nd311_solaris.tar`) under the previously created directory. You will see the following files:

   - `netdyn-setup`
   - `netdyn.tar.Z`
   - `jdk.tar.Z`
   - `install.README`
   - `release-notes-unix.html`

4. Follow the instructions in `install.README`, additionally considering the following instructions:

   Pre-installation checklist:

   - ad 1) Install NetDynamics as a Hyperwave system user
   - ad 3) Location of cgi-bin directory is `$DIRdcs/cgi/cgi-bin`

   A Hyperwave Information Server does not have a document directory like a normal web server because the documents are located in the Hyperwave database and not in the file system. Thus you have to create a temporary directory (e.g. `$HOME/nd11/doc`) which is specified as the web server's document directory during the installation process. A post-installation task will upload the content of this directory.

   Installation of NetDynamics:

   No additional instructions.

   Running the NetDynamics Application server:

   Configure correct environment: (tcsh)

- Make sure that the NetDynamics environment variables are set.

  Additional line in `.cshrc`:

  e.g. `source /usr2/users/hwsystem/nd311/NetDynamics20/bin/ndsetenv.csh`

- Make sure that the database environment is properly configured.

  Additional line in `.cshrc`:

  e.g. `setenv ORACLE_HOME /usr/app/oracle/product/7.3.2`

- Type the following:

  `source .cshrc`

  and the whole environment is set correctly.

Add the following line to `.db.contr.rc`

```
WAVEMASTER::CGI_PATH = cgi-bin/
```

Stop and start the server using the commands `hwstop` and `hwstart` so that the server is running under the correct environment.

For instructions on how to administrate the NetDynamics application server follow the instructions in the `install.README` file.

5. Post-Installation Tasks

Insert NetDynamics CGI object in Hyperwave, with e.g. the following attributes:

Attribute	Value
Type	Document
DocumentType:	CGI
Author	netdynamics
TimeCreated	97/06/13 09:17:51
TimeModified	97/07/16 08:13:53
Title	en:ndCGI.exe
Name	cgi-bin/ndCGI.exe
Path	cgi-bin/ndCGI.exe
GOid	0xc0a8991b_0x00011de0

The **Name** and **Path** attributes **must** have the same values as in this example.

Insert documents from temporary document directory:

Change to the previously created `doc` directory (`$HOME/nd311/doc`) and perform the following command:

```
$HOME/nd311/doc>hwupload -hwho <your Hyperwave host> -iden
netdynamics -par "~netdynamics"
```

3.7.3 CREATING YOUR FIRST NETDYNAMICS APPLICATION

See `http://www.netdynamics.com/developers/manuals/nd30/QuickStart/quickstart.htm`.

3.7.4 REFERENCING NETDYNAMICS PROJECTS

The extra path information is used to reference NetDynamics projects, e.g. suppose a project has the name *Project1* and starts with page *MasterPage*.

URL to start project:

`http://<Hyperwave_host>/cgi-bin/ndCGI.exe/Project1/MasterPage`

3.7.5 SHOWING HEADER AND FOOTER IN COMMON WITH NETDYNAMICS PROJECTS

If you would like to see the Hyperwave header and footer you have to add the attribute

MimeType=text/html

to the NetDynamics CGI object.

3.8 HOW TO CONNECT DATABASES TO HYPERWAVE USING NETDYNAMICS (NT)

3.8.1 REQUIREMENTS

To run NetDynamics you need Hyperwave version 2.5 or higher running on a system supported by NetDynamics. See `http://www.netdynamics.com`.

Download the appropriate NetDynamics package from `http://www.netdynamics.com/download/download.html`.

3.8.2 INSTALLING NETDYNAMICS

1. Log in to your NT system as system administrator
2. Installation process

Pre-installation checklist:

- add a NetDynamics user to the Hyperwave Information Server (e.g. a user with the name "netdynamics"). This step is recommended but not required.
- create a cgi-bin directory: $HOME/DCServer/cgi/cgi-bin (i.e. `c:\program files\Hyperwave\Server\DCServer\cgi\cgi-bin`).
- Hyperwave Information Servers do not have a document directory like web servers because the documents are located in the Hyperwave database and not in the file system. Thus you

have to create a temporary directory (e.g. `C:\programfiles\Hyperwave\Server\nd311\doc`) which is specified as the web server's document directory during the installation process. A post-installation task will upload the contents of this directory.

Installation:

- Start the program `nd311_windows.exe` and follow the instructions of the install program.

3. Configuration Tasks

Running NetDynamics Application server:

Configure correct enviroment:

- Make sure that the NetDynamics enviroment variables are set.

  NETDYN_HOME=C:\Spider\NetDynamics20\Projects

  NETDYN_HTML=C:\programfiles\Hyperwave\Server\nd311\doc

- Check the server's environment variable, which must point to the Hyperwave Information Server.

  HOME=C:\programfiles\Hyperwave\Server

- Add the following line to `.db.contr.rc`

  ```
  WAVEMASTER::CGI_PATH = cgi-bin/
  ```

Stop and start the server using the commands `hwstop` and `hwstart` so that the server is running under the correct environment.

To find out how to control the NetDynamics application server follow the instructions in the NetDynamics `install.README` file.

4. Post-Installation Tasks

Insert NetDynamics CGI object:

e.g. with the following attributes:

Attribute	Value
Type	Document
DocumentType:	CGI
Author	netdynamics
TimeCreated	97/06/13 09:17:51
TimeModified	97/07/16 08:13:53
Title	en:ndCGI.exe
Name	cgi-bin/ndCGI.exe
Path	cgi-bin/ndCGI.exe
GOid	0xc0a8991b_0x00011de0

Name and **Path** must have the same values as in this example.

Insert documents from temporary document directory:

Change to the previously created `doc` directory (`$HOME/nd311/doc`) and perform the following command:

```
$HOME/nd311/doc>hwupload -hwho <your Hyperwave host> -iden netdynamics
-par "~netdynamics"
```

3.8.3 CREATING YOUR FIRST NETDYNAMICS APPLICATION

See `http://www.netdynamics.com/developers/manuals/nd30/QuickStart/quickstart.htm` for the NetDynamics Quickstart.

3.8.4 REFERENCING NETDYNAMICS PROJECTS

The extra path info is used to reference NetDynamics projects, e.g., suppose a project has the name "Project1" and starts with page "MasterPage".

URL to start Project:

`http://`*<Hyperwave host>*`/cgi-bin/ndCGI.exe/Project1/MasterPage`

3.8.5 SHOWING HEADER AND FOOTER IN COMMON WITH NETDYNAMICS PROJECTS

If you would like to see the Hyperwave header and footer you have to add the attribute **MimeType** with the value "text/html" to the NetDynamics cgi object.

5. Installing the Hyperwave-NetDynamics DemoApplication

This is a modified version of the NetDynamics Project NetBusterVideo30Ph4. For further NetDynamics details visit the NetDynamics tutorial.

Example:

Step 1: insert the CGI object for the project.

Attribute	Value
Type	Document
DocumentType:	CGI
Author	netdynamics
Title	en:Hyperwave NetDynamics DemoDB
Name	cgi-bin/ndCGI.exe/HyperwaveDemo/pgLogin

You can set any value for **Title** but **Name** must point to the start page of your project.

Step 2: copy the project from the Hyperwave distribution directory *<cdrom>*`/3rdparty/netdyn/` to your NetDynamics project directory.

Note: *You need* `pkunzip` *or a compatible program.*

Change directory to the NetDynamics project directory (assuming your projects are in `c:\spider\NetDynamics20\Projects\`)

```
cd c:\spider\NetDynamics20\Projects\
```

```
tar -xf <cdrom>/3rdparty/netdyn/hwdemo.tar
```

Step3: for every Hyperwave user who is granted access to the demodatabase add a user entry in the `ndVideo.mdb` CUSTOMER table (with Microsoft Access) and for each of these users add the access rights in the WEB_USERS table. This is found under `c:\spider\NetDynamics\Projects\databases\ndVideo.mdb`.

Note: *The customer_password is ignored because every Hyperwave user is already identified when he tries to access the database so you don't need to maintain this field.*

Note: *The customer_id (customer table) and the user_id (Web_Users table) must match your Hyperwave account name.*

When you are finished, to start the project click on the new CGI object. The Hyperwave server and the NetDynamics application must be running when you do this.

3.9 THE HYPERWAVE GATEWAY INTERFACE (HGI)

HGI was developed to provide a powerful way to integrate external databases in Hyperwave. See the *Hyperwave Programmer's Guide* for details.

3.10 HYPERWAVE SERVER POOL

- The Hyperwave Server Pool is a tool used to group servers together so that they function like a single server, e.g. links between the servers are always kept consistent, and it is possible to set up a central user database.

3.10.1 CONFIGURING A SERVER POOL

To construct a Hyperwave Server Pool you have to perform the steps described here. First, the members are specified in a file which contains important information about each pool server. The required attributes in the file are:

- **UServer**: IP address of the pool server
- **OHName**: The server's official hostname, that is its primary name as a qualified domain name.
- **SString**: A string which contains a readable description of the pool server.

Each pool server entry is separated by a blank line.

Example file:

```
UServer=129.27.2.5
OHName=info.tu-graz.ac.at
SString=Information System of Graz University of Technology


UServer=129.27.153.30
OHName= fiicmss02.tu-graz.ac.at
SString= AEIOU Information Server
```

After you have taken the steps above, you must use the pool manager tool `hwpoolmanager` (see below) to create the pool. You can optionally configure a master server for global identification before activating the tool.

3.10.2 HWPOOLMANAGER

When you are finished with the steps above you can execute the Hyperwave tool `hwpoolmanager`. This is done by entering

`hwpoolmanager -file` *filename*

in the command line, where *filename* is the name of the pool file.

The tool contacts each server listed in the file and performs the following tasks:

- creates the base structure each pool server needs
  - collection "Hyperwave Server Pool" (child of root collection)
  - collection "Update"
  - collection "Send Update Request"
- creates a server object for each pool server with the exception of the local server.

Note: *To build a Hyperwave Server Pool you need system rights for each server in the pool. If you are not auto identified as a system user the tool asks for identification.*

The next step is to make some changes in the `.db.contr.rc` (main configuration file) of each pool server. Add the line:

```
HGSERVER::POOLUPDATE = TRUE
```

which tells the server that it is a pool server.

Configure the pool update process with

```
MAIN::PROCESS = POOLUPDATE
POOLUPDATE::LOG = poolupdate.log
POOLUPDATE::COMMAND = hwpoolupdate -logfile $LOG -sleep 120
```

The sleeptime is the time between two update cycles

See also Configuring the Server with .db.contr.rc on page 14.

3.10.3 ACTIVATING THE SERVER POOL

The last step when starting the server pool is to perform `hwstop` and `hwstart` (see *Hyperwave Installation Guide*) on each pool server so that the server recognizes the changes in `.db.contr.rc`. You will then have a running Server Pool.

3.10.4 CHANGING YOUR SERVER POOL

If you want to add or remove a server from the pool you have to perform the following steps:

1. Edit your pool file. Add new servers or delete the entry for the server to be removed.
2. Execute `hwpoolmanager`. This tool contacts each server and updates the pool structure. If a server is deleted `hwpoolmanager` destroys the pool on this server.
3. Edit the `.db.contr.rc` file of the new or removed pool servers. If a server is new make the same changes as above. If a server is deleted you have to remove its pool server entries.

4. Perform `hwstop` and `hwstart` on servers which are new or deleted.

3.10.5 SETTING UP GLOBAL IDENTIFICATION

The server pool has the optional feature of global identification, i.e. the pool can be configured so that when a user logs in to a server in the pool, the user is identified automatically on all servers in the pool.

***Note**: If you have a server pool but have not configured global identification, then in most cases you will only be able to log in to the local server. However, if there is a user in the user database of a remote server with the same user name and password as the local user you are logging in as, then you will also be logged into that remote server.*

When configuring global identification in a server pool, one server in the pool must be selected as *master server*. This may, for example, be a server which has a gateway to an external user data base.

The following entries must be made in `.db.contr.rc` for the master server:

```
MAIN::PROCESS = HW_IDENTD
HW_IDENTD::LOG = $HOME/log/hw_identd.log
HW_IDENTD::COMMAND = hw_identd
```

***Note**: The* `hw_identd` *module is not available for the Windows NT platform. Thus the master server (or at least the* `hw_identd` *process) must run on a UNIX platform.*

For all the servers in the pool except the master, entries like the following must be made in the main server configuration file `.db.contr.rc`:

```
HGSERVER::AUTHENTICATION[local]
HGSERVER::AUTHENTICATION[hw_pool] = TCP orion 4567
```

The first line tells the server to attempt to identify a user who is logging in by using the local data base of that server. The second tells it to connect to the master server to try to find the user name in any user data bases it has access to. The order of these two lines determines which method is tried first. In the second line, the part `TCP orion 4567` consists of the protocol, the master server name and the port number respectively.

***Note:** Suppose the pool system user and a local system user have the same user name and password (e.g.* `hwsystem`*) and global identification has been configured. If you log in as user* `hwsystem` *with the local password, you will be the local system user, but you will be an anonymous user on the other servers in the pool. If you log in as user* `hwsystem` *with the global password, you will be the system user on the whole server pool.*

Global identification is activated the same way the server pool is activated (see page 77).

3.10.6 SERVER POOL FEATURES

Starting with version 2.6 of Hyperwave, it is possible to run a search on the entire server pool (see the *Hyperwave User's Guide*).

***Note:** For the search function to work correctly when searching in a server pool, all servers in the pool must have the same fulltext engine switched on, that is, either all servers should use the native search engine or all should use Verity.*

Further server pool features are the option to link documents and collections from remote servers to your local server and to create links to remote documents, collections or destination anchors. This is done in the same way as for documents on the same server (see the *Hyperwave User's Guide*)

3.11 WAVEMASTER SECURITY

A version of WaveMaster with SSL (secure socket layer) support is available. When using it, the information sent to (and from) a Web client is encrypted, and the server is authenticated, i.e. the client can be reasonably sure that the server is who it says it is.

To get WaveMaster with SSL, contact Hyperwave Information Management GmbH (`support@hyperwave.com`).

Note: *WaveMaster with SSL is not yet available for the Windows NT server.*

4 DOCUMENT MANAGEMENT

Hyperwave has a variety of command line tools which can be used to retrieve, insert and manipulate information. The tools can be used by themselves or as part of a script.

4.1 USING THE WINDOWS NT COMMAND LINE TOOLS

All the tools in this chapter are available for both UNIX and Windows NT. Note that the tools can also be run under Windows 95.

In all the descriptions of command line tools found in this chapter, it is stated that when using arguments which consist of more than one word simple quotation marks (') are used, which is only the case for UNIX tools. Please note that for the NT tools, arguments consisting of more than one word must be enclosed in double quotation marks ("), e.g.

```
hwinfo -coll gotest -mult -iden "hwsystem hwsystem"
```

4.2 GENERAL INFORMATION ABOUT THE COMMAND LINE TOOLS

Note that it is not always necessary when entering an option for a command line tool to enter the entire option. Only a unique prefix need be entered, e.g. `–mult` can be used instead of `–multiple`.

4.2.1 DEFAULT VALUES AND ENVIRONMENT VARIABLES

Some options have default values and/or can be set by enviroment variables. Default values are overridden by enviroment variables which are overridden by command line options. The following table summarizes the environment variables available and the corresponding command line options and default values.

Option	Default Value	Enviroment Variable	Short Description
-hwhost	localhost	HWHOST	name of the Hyperwave server
-hwport	418	HWPORT	port number of the Hyperwave server
-language	en	HWLANGUAGE	language of the title
-sortorder	ATC	HWSORTORDER	sort order of objects
-cdate	current time	HWCDATE	creation time
-odate	-	HWODATE	opening time
-edate	-	HWEDATE	expiration time

-rights	-	HWRIGHTS	access rights
-user	-	HWUSER	identify as user
-parent	-	HWPARENT	name or id of parent collection

4.3 UPLOADING AND DOWNLOADING DIRECTORIES AND DOCUMENTS

If you have a WWW server or directories which contain files you would like to upload to Hyperwave or if you want to download the contents of a Hyperwave Information Server to a WWW server or file system, you can use the `hwupload` and `hwdownload` tools.

4.3.1 HWUPLOAD

When using `hwupload` to upload files, the directory structure is converted to a collection hierarchy, so that directory `/path/dirname` is mapped to a collection named "path/dirname", which is a sub-collection of collection "path". The root of the directory structure you are uploading is mapped to a collection you specify, and which must have been created before running `hwupload`. The same procedure applies to documents, which automatically get the name "/path/filename".

As can be seen above, the `hwupload` utility does not structure your information in a new way, but rather just takes your WWW directory tree and converts it to a Hyperwave collection hierarchy. This basic structure is created automatically and very quickly, but it may not be a particularly good hierarchy. It might be a good idea to use automatic conversion to test if Hyperwave can handle information in a way you are satisfied with. You may decide to create a well-designed collection tree by hand or modify the automatically created one later on as the need arises.

In Hyperwave, URLs are formed as follows: the server name is followed by a slash and then the collection or document name, e.g. `http://www.hyperwave.com/hyperwave` to refer to a collection named "hyperwave". Thus URLs in Hyperwave have the same appearance as WWW URLs. Because paths are mapped to corresponding collection names, the same URLs are valid after you have converted to Hyperwave as were previously valid. This means that existing URLs (for example, stored in user's hotlists) remain valid when moving to a Hyperwave Information Server, provided that the hostname remains the same.

Internal links (those pointing to documents on the same server) are stored within the link database and consistency is maintained automatically. External links (pointing to other servers, and non-HTTP URLs) are also stored in the link database, but without guaranteed consistency, as is the case with any other WWW server.

`hwupload` sets the attribute 'PresentationHints=FullCollectionHead' for welcome, index or home HTML files. This affects the presentation of these files in the following way: by default when a collection is accessed, it displays only a list of its members, but when one of the members has been designated as a full collection head, it is automatically displayed when a collection is accessed and the list of members is not shown.

RUNNING HWUPLOAD

The `hwupload` tool is run by entering `hwupload` *options* in the command line. The options and their possible values are explained below.

GETTING HELP AND VERSION INFORMATION

Use `-help` to print the usage information (a short description of each option) and the default values and `-version` to print the version string of the tool. `-hwhost` *hwhost* specifies the name of the Hyperwave host and overrides the environment variable HWHOST.

IDENTIFYING

Identification is done with `-identify ['`*user*`[` *password*`]']`. If, for example, you only enter the user name, you will be prompted for the password. To shift the identification to a different user, use `-user` *user*, which overrides HWUSER. This last option is only available to system users.

SPECIFYING THE COLLECTION WHERE YOU WANT TO UPLOAD THE OBJECTS

The option `-parent` *parent* is used to specify the collection where the WWW directory should be uploaded, where *parent* is the name or object id of the collection. The default is '/' (the root collection) and the corresponding environment variable is HWPARENT. `-language` *lang* sets the language of the **Title** attribute to *lang*. Objects without the `<title>` tag get file names as their **Title** attribute.

There are several options pertaining to the selection of the server and directory to upload.

`-hopo` *hostport*

host name and port (default 80) of the WWW server being loaded

Examples: `-hopo www.iicm.edu:8000` specifies a server with port 8000 and `-hopo www.iicm.edu` specifies the server and the default port.

`-basedir` *basedir*

path to the WWW server's '/' directory (default is '`.`')

`-startdir` *startdir*

lets you designate a subtree of the web server to upload. The default is *basedir*. **Note**: if you use `-startdir`, make sure that `-basedir` is correctly set so that `hwupload` can correctly interpret relative links.

`-nonrecursive`

inserts only the contents of *startdir*

`-nolinks`

doesn't follow symbolic links through the file system

OPTIONS FOR DOCUMENT TYPES

The options below allow you to select which document types you want to upload. These options don't work if you use the `-magic` option.

`-hmc`

inserts only HM-Card documents (files with extension `.hmc`)

`-html`

inserts only HTML documents (files with extension `.html` or `.htm`)

`-image`

inserts only image documents (files with extension `.bmp`, `.tif`, `.gif`, `.jpg`, `.xpm`, `.ppm`, `.xbm`, `.pbm`)

`-java`

inserts only JAVA classes (files with extension `.class`)

`-movie`

inserts only movie documents (files with extension `.mpg`)

`-ps`

inserts only PostScript documents (files with extension `.ps`)

`-sound`

inserts only sound documents (files with extension `.wav`, `.au`)

`-txt`

inserts only ASCII documents (files with extension `.txt`, `.java`)

`-other`

prints names of all files that could not be inserted because of unknown types

`-nodot`

ignores directories and filenames beginning with '`.`'

`-addtype`

adds new type to insert (extension|type|mimetype)

`-magic`

does magic number test to detect document types

WHAT TO DO WITH EXISTING DOCUMENTS

There are two options which you can use to tell `hwupload` what to do when it tries to upload an object which is already on the server. If you use `-replace`, it will simply overwrite existing documents. With `-exists`, it will not replace any existing documents. With the `-nochanges` option, no insertions take place and directories are scanned only.

.HMI FILES

The option `-hmi` is used to tell `hwupload` to use `.hmi` files to recreate attributes for Hyperwave objects that were downloaded using `hwdownload`. See Using hwupload and hwdownload Together on page 85 for more details.

`hwupload` can optionally create a log file with the option `-logfile` *logfile*, in which case it writes all comments and messages to a file instead of to stdout. Statistics (accumulated logging information) can be created using `-statistics`. The statistics are appended to the log file, if given, and otherwise go to stdout.

INHERITING RIGHTS

The option `–inheritrights` causes the uploaded documents and collections to inherit the rights attribute from the collection they are inserted into. This option does not work if `.hmi` files are used.

PREFIX AND FULLTITLE OPTIONS

`-prefix` *prefix*

This option makes *prefix* the prefix of all the names of objects which are uploaded, whether the names are from `.hmi` files or are generated by `hwupload`.

`-fulltitle`

This option causes the full directory path to be used as the collection title.

VERSION CONTROL

`hwupload` uses the options `-checkout`, `-commitversion` and `-forcedcheckin` to handle version controlled documents or cause new documents to be version controlled. See page 93.

UPLOADING A TAR FILE

`-tarfile` *filename*

This option lets you upload the specified tar file.

EXAMPLES

```
hwupload -identify willi -parent "~willi" -basedir documentation
```

This command is used to upload an entire directory tree recursively starting at directory `documentation` to the collection named "~willi". Each document gets its directory path (starting at directory `documentation`) as **Name** attribute. Note that the collection name must be escaped by quotes so that the "~" is not interpreted incorrectly.

```
hwupload -parent reports97 -basedir reports -nonrecursive -txt
```

This command uploads only the text files contained directly in the directory `reports` to the collection "reports97". Because the `-identify` option is not used, this won't work unless the user can be automatically identified.

4.3.2 HWDOWNLOAD

`hwdownload` does the opposite of `hwupload`, i.e. it takes the collection structure and maps it to a corresponding directory structure.

RUNNING HWDOWNLOAD

`hwdownload` is run by entering `hwdownload` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost* (default localhost), `-identify ['`*user*`[`*password*`]']`, and `-user` *user* work the same way as above for `hwupload`. Another option usable with `hwdownload` is `-hwport` *port*, which lets you specify the port the Hyperwave Information Server is listening to (default 418). Identification is, of course, only required when downloading restricted documents. However, it is a good idea to always identify when using this tool because if you are downloading a collection that contains any restricted documents, those documents will not be downloaded along with the other unrestricted ones.

SELECTING THE COLLECTION OR OBJECT YOU WANT TO DOWNLOAD

The options `-parent` *parent* and `-object` *object* are used to select the Hyperwave objects you want to download. In either case the value is the name or object id of the object or collection. The `-parent` option allows you to select a collection whose (recursive) children you want to download without downloading the collection itself. The default value of `parent` is "/", Hyperwave's root collection. With `-object` you can specify one particular document or you can select a collection. If you select a collection then the collection as well as its recursive children are downloaded.

You can tell `hwdownload` the path to download data into using `-targetdir` *targetdir*. Default is "`.`" (the current directory). This directory must exist (it will not be created with this option).

If you want to strip a prefix from the names of the objects you are downloading, use `-stripprefix` *stripprefix*. For example, if you have a document with the name "~hwuser/report.html", and you enter the prefix `~hwuser/`, then the document will end up with the name report.html.

The option `-base` *base* lets you specify the base to create in exported HTML files, e.g. `http://my.new.server:815/`.

HANDLING LINKS WHICH ARE OUT OF SCOPE

The option `-outofscopetarget` *target* is used to handle links which originate in documents being exported but whose destinations are in documents on the Hyperwave server which are not being exported. The value of *target* is a URL. If not set, these links will be removed. It allows you to point the links to a Hyperwave gateway or already downloaded collections.

The use of the option `-nonrecursive` lets you specify not to handle collections recursively. If, for example, you use the options `-parent` *parent* and `-nonrecursive`, you will download only the documents contained directly in that collection. By default, `.hmi` files (files which contain the attributes of the object) are created for every object you download. The option `-nohmi` prevents the creation of these files. Normally if one collection is contained in two or more others, the collection is physically downloaded to one directory and the other directories just have links to it. The `-nolinks` option tells `hwdownload` not to create any file system links, thus losing some collection relation information.

EXAMPLES

The most simple application of `hwdownload` looks like this:

`hwdownload -object` *someobject*
or

`hwdownload -parent` *somecollection*
This recursively downloads an object and its children (if there are any) in the first case, and in the latter case the children of a collection.

```
hwdownload -hwho someserver -ident myname -object someobject -
targetdir wheretodir -outofscopetarget http://someserver
```

This is a more typical example. This command downloads a collection tree to the file system. `someserver` is the Internet address of the Hyperwave Information Server containing the collection (or other object) `someobject`, which is to be downloaded recursively. If objects with restricted access rights are downloaded, then identification (`myname`) is necessary. `-targetdir` is used to download the information to a directory other than the current one. In order to keep links to documents which are on the server but outside the scope of the currently downloaded collection tree, the base URL `http://someserver` is specified in order to make these links point to the gateway of the server at that URL.

To download the collection named "just_for_fun" on the server `www.someplace.org` to the directory `~myhome/fun` where the collection can only be seen by identified users, a command such as the following can be used:

```
hwdownload -hwho www.someplace.org -ident you -object
just_for_fun -targetdir ~myhome/fun -outofscopetarget
http://www.someplace.org
```

If you want to download a collection to be used by a standard web server you have to specify the relative base of the documents on that web server. This can be done with the following command:

```
hwdownload -hwho www.someplace.org -ident you -object
just_for_fun -targetdir ~myhome/fun -outofscopetarget
http://www.someplace.org -base
http://www.standardweb.server.com/
```

And all links will be created accordingly.

4.3.3 USING HWUPLOAD AND HWDOWNLOAD TOGETHER

WORKING WITH .HMI FILES

There are certain things you should consider if you are downloading documents from a Hyperwave Information Server and plan to upload them to Hyperwave later. `hwdownload` produces `.hmi` files by default when downloading from a Hyperwave Information Server. These files contain the attributes that are associated with the corresponding server object and one such file is created for each object exported. It is recommended that you do not suppress the creation of these files with the `-nohmi` option so that when the files are uploaded to Hyperwave again, their attributes can be recreated. By default `hwupload` does not recreate the attributes. You must use `hwupload`'s `-hmi` option if you want to restore the attributes.

4.3.4 INSERTING ATTRIBUTES WITH HWUPLOAD

As mentioned above, `.hmi` files can be used to download and upload attributes. However, it is not necessary to use only the automatically produced files which are created when you download from a Hyperwave Information Server. If you download from a Web server no `.hmi` files are produced of course, but you may want to give the objects you downloaded attributes when uploading them to Hyperwave. To do this, you have to create your own `.hmi` files.

.HMI FILE SYNTAX

The syntax of these files can be described as follows:

```
HMI := ID Separator Object

Separator := NL

ID := 'HyperWave Meta Information/' MajorVersion '.'
MinorVersion NL
```

```
Object := { FieldName '=' FieldValue NL}*
MajorVersion := {'0' .. '9'}*
MinorVersion := {'0' .. '9'}*
```

All this means is that the file consists of a string which gives the version number of the `.hmi` file format (currently 1.0), followed by a blank line, followed by any number of attribute-value pairs. An example of this is the following:

```
HyperWave Meta Information/1.0

ObjectID=0x000052fd
Type=Document
DocumentType=collection
Author=system
TimeCreated=97/02/10 13:06:26
Title=en:Test Collection
Name=~vmayr/test
Subdocs=8
GOid=0xc0a89919 0x000052fd
```

The attributes can be listed in any order you wish. Also, keep in mind that when you try to enter an attribute this way, it will not work if e.g. you try to give an object a name which already exists on the server, or give it a type which isn't the same as the actual type.

NAMING .HMI FILES

The files must be named such that they consist of the file name of the corresponding object with the extension `.hmi` added on, e.g. if the file name is `pict1.jpg`, then the `.hmi` file must be called `pict1.jpg.hmi`.

4.4 MIRRORING HYPERWAVE DATA

`hwmirrorout` and `hwmirrorin` are two tools which are used to mirror data from one Hyperwave Information Server to another. They are described below.

4.4.1 HWMIRROROUT

`hwmirrorout` can be used to map the structure and contents of one or more directories to a corresponding file structure, or store them in a tar file. This downloaded data can then be used for mirroring purposes.

To prevent sending large amounts of data to update a mirror, `hwmirrorout` allows you to build an incremental mirror. Thus only the structure information and new or changed data are included. In order to do this, `hwmirrorout` creates a lookup file (`mirror.lkp`) containing a time-stamp recording when the file was built and information about the content.

Note: *Only links which have destinations within the mirror structure are downloaded, but it is possible to specify as many objects as you like.*

`hwmirrorout` has the following options:

`-help`, `-version`, `-hwhost`, `-identify` ['*user*[*password*]'], and `-user` *user* work the same way as for the other tools (see page 82).

`-object` *object*

Use this option to specify one or more objects/collections you want to include in your mirror using their **Name** attribute or object id.

`-incremental`

Specifies that an incremental mirror should be built. See section How to Create a Mirror.

`-targetdir` *targetdir*

This option is used to specify the path to the directory you want to download the data to.

`-tarfile` *name*

Name of tar file to be created.

EXAMPLE

```
hwmirrorout -identify -hwhost intranet.xfone.com -object
'~misc' -incremental -targetdir c:\hyperwave\mirror
```

Makes an incremental mirror of the contents of collection "~misc" on the server `intranet.xfone.com` to the directory `c:\hyperwave\mirror`.

4.4.2 HWMIRRORIN

`hwmirrorin` can be used to insert or update mirror data on a Hyperwave Information Server.

It uses the following options:

`-help`, `-version`, `-hwhost`, `-identify` ['*user*[*password*]'], and `-user` *user* work the same way as for the other tools (see page 82).

`-parent` *parent*

The **Name** attribute or object id of the parent collection of the mirror.

`-basedir` *basedir*

The starting point of the mirror in the directory structure.

`-tarfile` *tarfile*

The name of the tar file to mirror.

`-prefix` *prefix*

Prefix to be given to all **Name** attributes found in the mirror file(s).

`-author`

If this option is used, the original author name is inserted for the objects instead of the name of the user who inserts the mirror (option for system users only).

`-logfile`

Create log file.

`-strict`

This option causes `hwmirrorin` to create an exact copy of the structure found in the mirror file. Thus it deletes all child collections and child documents of the specified parent (option `-parent`) which are not found in the mirror file. This means that if you accidentally try to mirror a collection to a completely different collection when using this option, the contents of that collection will be deleted. Use caution when using this option, especially when making a mirror in the root collection.

DEFAULT BEHAVIOR OF HWMIRRORIN

The default behavior of `hwmirrorin` is not to overwrite existing objects in the collection which the mirror is being uploaded to unless they correspond to the collection being mirrored. The tool looks for a collection that corresponds to the collection to be mirrored among those directly contained in the collection given by the option `-parent`. If the tool does not find a collection which matches the mirror, it simply adds the mirror to the other collections which are contained in

the parent collection. If it does find a corresponding collection, it mirrors to that collection and leaves the other collections in the parent untouched. Thus there is no danger that you will delete objects from the collection hierarchy accidentally. This behavior can be overridden using the option `-strict` (see above).

EXAMPLE

```
hwmirrorin -identify -parent software/documentation -hwho
www.somedomain.com -tarfile docu.tar -prefix
software/documentation
```

Uploads the mirror in the tar file `docu.tar` to the collection "software/documentation" on the server `www.somedomain.com`. All mirrored documents get the prefix "software/documentation" added to their **Name** attributes.

4.4.3 HOW TO CREATE A MIRROR

1. Download the parts of the Hyperwave Information Server you would like to mirror to a directory structure or to a tar file with `hwmirrorout`. The tool creates a mirror file (`mirror.lkp`).
2. Upload the data to one or more remote servers. Note that it is not possible to upload more than one instance of a mirror object to a server. This is because the server stores a unique *repl. ID* (which is the GOid of the object that was mirrored) for each mirror object.
3. To create incremental updates download the same parts using `hwmirrorout` with the option `-incremental`. Note that `hwmirrorout` needs the file `mirror.lkp` to build an incremental mirror. If this file is not available, a warning message appears ("can't open lookup file mirror.lkp") and `hwmirrorout` builds a complete mirror.

   When making a **mirror in a tar file**, an incremental mirror file must be given a new name because already existing mirror files are not overwritten.

   When making a **mirror in the file system**, delete the previous mirror (but do not delete `mirror.lkp`) or copy `mirror.lkp` to another empty directory and use it as the base of your incremental mirror.
4. Update mirror(s) using `hwmirrorin`.

4.5 INSERTING DOCUMENTS

There are three command line tools available for inserting documents and collections into Hyperwave. These tools and their options are described below.

4.5.1 HWINSCOLL

`hwinscoll` is used to insert collections and clusters in Hyperwave. It is run by entering `hwinscoll` *options* in the command line. Some options, e.g. `-help`, are used alone, while others (those which are shown with a text in italics, e.g. `-hwhost` *hwhost*) must be followed by a value. Also, in some cases there are environment variables which can be set instead of using the options. All options are explained below.

If you want help or version information, use `-help` to print the usage information (a short description of each option) and the default values, or `-version` to print the version string of this tool.

To specify which server you want to create the collection(s) or cluster(s) on, you may use `-hwhost` *hwhost*. This overrides the HWHOST environment variable. The option `-hwport` *hwport* specifies the port number to connect to and overrides the HWPORT environment variable.

To identify yourself manually, use `-identify` [`'`*user* [*password*]`'`]. System users can also use the option `-user` *user*, where *user* is a valid user name for an account on the server where you are inserting the collection. The `-user` option shifts the identification from your account to *user*'s account. Thus if you use it with `hwinscoll`, the collections you insert will have the owner `user`.

An option which is absolutely required when using `hwinscoll` is `-parent` *parent* which is used to specify the collection you want to insert the collection or cluster into. The value of *parent* is the unique name or object id of the desired collection. This overrides the environment variable HWPARENT.

NAMING COLLECTIONS

In Hyperwave, collections are required to have a name which is unique on the server and thus the `-name` *name* option exists in order to specify this name. It is also possible to set several names at once. For example, `-name n1 -name 'n2 n3 n4' -name n5` sets the name attributes n1, n2, n3, n4 and n5. The `-name` option is not required if you are inserting a cluster (see below).

GIVING THE COLLECTION A TITLE

You must specify the title of the collection using the option `-title` *title*. The `-language` *lang* option is optional (default is English) and is used to specify the language of the title. The value of *lang* must be one of Hyperwave's language prefixes. These are as follows:

LANGUAGE ABBREVIATIONS

Language	Abbreviation
English	en
German	ge
French	fr
Italian	it
Spanish	sp
Japanese	jp

This option overrides the HWLANGUAGE environment variable. It has no effect for a formatted title (see next paragraph).

Formatted titles allow you to give an object extra attributes that cannot be specified as separate options. The attributes are entered as *attribute_name=value* and must be separated by a backslash ("\"). When using a formatted title, the option `-formatted` must be used.

Examples:

To insert a title and a keyword, enter

```
-formatted -title 'Title=en:title\Keyword=keyword1'
```

among the options for `hwinscoll`.

To insert titles in two different languages, enter

```
-formatted -title 'Title=en:Welcome\Title=ge:Willkommen'
```

INSERTING CLUSTERS

To insert a cluster, use the `-cluster` option. Since clusters don't require a name, `-name` is not necessary in this case.

FURTHER OPTIONS

It is possible to specify separately some (not all) attributes for the object you are inserting. Below is a description of these options and their values along with the environment variables they override (written in capital letters).

`-cdate` *cdate* (HWCDATE)

sets creation time

`-odate` *odate* (HWODATE)

sets opening time

`-edate` *edate* (HWEDATE)

sets expiration time

`-rights` *rights* (HWRIGHTS)

sets the access rights of the collection

`-description` *description*

adds a short description

`-sortorder` *so* (HWSORTORDER)

defines the order in which the collection's children should be displayed

The values for `-cdate`, `-edate` and `-odate` are dates, which are entered in the form "yy/mm/dd hh:mm:ss", e.g. `97/01/13 16:32:15` for the date January 1, 1997 at 4:32 p.m. and 15 seconds. You can also go only to the desired accuracy if you wish, e.g. you can enter `97/04/05` and leave out the exact time. For access rights see page 63.

EXAMPLES

```
hwinscoll -hwhost www.myserver.com -parent doc -name doc/nov96
-title 'November 1996'
```

This inserts a subcollection in collection "doc" with name "doc/nov96" and title "en:November 1996" on the server `www.myserver.com`.

```
hwinscoll -hwhost intranet.xyz.com -parent '~fmaier' -form -
title 'Title=en:Welcome\Title=ge:Willkommen' -edate 97/04/01 -
cluster
```

This inserts a multilingual cluster with an expiration date on the server `intranet.xyz.com`.

4.5.2 HWINSDOC

`hwinsdoc` is used to insert or replace any type of Hyperwave document, e.g. images, movies, generic, CGI, remote objects with HTTP, FTP or Telnet protocols, etc.

RUNNING HWINSDOC

`hwinsdoc` is run by entering `hwinsdoc` *options* in the command line. Some options, e.g. `-help`, are used alone, while others (those which are shown with a text in italics, e.g. `-hwhost` *hwhost*) must be followed by a value. Also, in some cases there are environment variables which can be set instead of using the options. All options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` ['*user* [*password*]'] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

REPLACING DOCUMENTS

There are two options with `hwinsdoc`: you can replace an existing document, or you can insert a new one. There are two command line options which are used when replacing documents: with `-replace` *object* you specify the object on the server you want to replace, using its name or object id. The option `-path` *path* tells `hwinsdoc` where to insert the new document. The value of *path* should be the name or object id of the parent collection. This option overrides the HWPARENT environment variable. Another required option is `-type` *type* which sets the **DocumentType** attribute for the document. The value of *type* must be one of the following:

INSERTING NEW DOCUMENTS

Type	Abbreviation
Remote	R
Program	P
CGI	C
Generic	G
Image	I
text	T
Movie	M
Sound	S
Scene	3D
PostScript	P

SETTING THE MIME TYPE

Option `-mime` *mimetype* sets the **MimeType** attribute of the document. This value is checked against the value of option `-type`. If option `-type` is omitted, `hwinsdoc` tries to generate the **DocumentType** attribute from the MIME type given.

Note: **DocumentType** *"text" sets the* **MimeType** *attribute to "text/plain". When inserting HTML documents you must set the* **MimeType** *attribute with option* `-mimetype text/html`. *In this case you can omit the option* `-type`.

The `-title` *title* option is also required unless you are inserting HTML documents. Otherwise, the options `-title` *title*, `-language` *lang* and `-formatted` are used in the same manner as described above for `hwinscoll` (see page 89).

The `-name` *name* option is used to give objects a name. Unlike collections, documents do not require names, but if you assign a name, then it must be unique on the server. As for `hwinscoll`, the option may be used to give multiple names (see page 89).

It is also possible to specify certain other attributes. These are the creation time (`-cdate` *cdate*), opening time (`-odate` *odate*), expiration time (`-edate` *edate*) and access rights (`-rights` *rights*). For information on how to use these options, see page 89.

To set attributes that you cannot set directly with the options (e.g. **Keyword**), use the option `-attributes` *file*. The file must consist of lines of the form *attribute=value*. `hwinsdoc` stops reading on empty lines or end of file.

OPTIONS WHICH DEPEND ON THE TYPE OF DOCUMENT BEING INSERTED

The options which are dependent on the type of document being inserted are described below. You have to use one of these options; and it must be in accordance with the type (MIME type) set. If only one option is available then it is required. Generic documents require the option `-path` and remote documents the option `-protocol`.

When inserting image, movie, PostScript, program, scene, sound or text (plain or HTML) documents the `-path` *path* option specifies the name of the file which contains the data of the new document.

When inserting a CGI document, `-relurl` *relurl* specifies a reference to a CGI program reachable via the CGI directory of the server (`~hwsystem/dcserver/cgi`).

When inserting a document of type 'Generic', `-path` *path* specifies the name of the file which contains the data of the new document. `-genmime` *mimetype* sets the mime type of the generic document and `-arguments` *arguments* sets the arguments of the generic document.

When inserting a remote document, `-protocol` *protocol* sets the protocol and thus the type of remote document. `-host` *host* sets the remote host (e.g. WWW host). `-port` *port* sets the remote port (e.g. WWW port) and `-path` *path* sets the path. What the path is depends on the protocol type (e.g. path of the WWW document).

The option `-test` prints only the attributes of the document and does not insert the document.

VERSION CONTROL

`hwinsdoc` uses the options `–checkout`, `-commitversion` and `–forcedcheckin` to handle version controlled documents or cause new documents to be version controlled. See page 93.

SETTING ATTRIBUTES FROM WITHIN HTML FILES

When inserting an HTML text you can give it certain attributes by entering them directly in the HTML file. This is done using the NAME attribute of the META tag.

Examples:

```
<META NAME="TimeOpen" CONTENT=ODate>
```

Sets the **TimeOpen** attribute for this document.

```
<META NAME="Keyword" CONTENT=Keyword>
```

Sets a **Keyword** attribute for this document.

```
<META NAME="abc" CONTENT=123>
```

Sets the user defined attribute **abc** for this document.

EXAMPLES

```
hwinsdoc -hwhost host.mydomain.com -parent testcoll -mime
text/html -path mytext.html
```

This inserts the HTML text `mytext.html` in collection "testcoll" on the server `host.mydomain.com`.

```
hwinsdoc -hwhost intranet.xfone.com -par hotlist -title
'Introducing HTML 3.2' -rights 'R:a' -type R -prot http -host
www.w3.org -path '/pub/WWW/MarkUp/Wilbur/'
```

This inserts the URL of the introduction to HTML 3.2 into a collection with name "hotlist" on the server `intranet.xfone.com`. The rights are set such that the remote object is only be readable for its author.

```
hwinsdoc -hwhost host.abc.com -par pictures -title 'Car' -type
I -pat car.gif
```

This inserts the image file `car.gif` into collection "pictures" to the server `host.abc.com`.

```
hwinsdoc -hwhost host.domain.com -replace mytext -path
mytextnew.html
```

Replaces the document with name "mytext" on the server `host.domain.com`.

4.5.3 HWINSTEXT

`hwinstext` may be used to insert several HTML documents at once. Any file given in the file list may contain more than one HTML document. It is also possible to replace an HTML document (this can only be done one at a time).

RUNNING HWINSTEXT

`hwinstext` is run by entering `hwinstext` *options* in the command line. Some options, e.g. `-help`, are used alone, while others (those which are shown with a text in italics, e.g. `-hwhost` *hwhost*) must be followed by a value. Also, in some cases there are environment variables which can be set instead of using the options. All options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

REPLACING TEXTS

With `hwinstext` you have two options: you can replace an existing text, or you can insert a new one. There are two options which are used when replacing texts: with `-replace` *document* you specify the text you want to replace, using its name or object id. The option `-file` *file* tells `hwinstext` where the file which you are replacing the old text with is located. Only one file name is allowed.

INSERTING NEW TEXTS

Some options are required, e.g. `-parent` *parent*, which tells `hwinstext` where to insert the new document. The value should be the name or object id of the parent collection. This option overrides the HWPARENT environment variable. Another required option is `-file` *file* which is used to specify the names of the files which contain the document data. It is possible to specify several file names at once, e.g. `-file f1 -file 'f2 f3 f4' -file f5` specifies files f1, f2, f3, f4, and f5. The only time this is not required is when documents are piped into `hwinstext`.

The `-name` *name* option specifies the name attribute of the first document. It is possible to use it to set several names at once, e.g. `-name n1 -name 'n2 n3 n4' -name n5` sets the name attributes n1, n2, n3, n4 and n5. The language of all documents can be set using the `-language` *lang* option, which overrides the HWLANGUAGE environment variable. The value of *lang* is a Hyperwave language prefix, which is a two-letter abbreviation for a language available in Hyperwave. The language prefixes are given in the section on `hwinscoll` on page 89.

When inserting HTML documents, the option `-html` must be used. `-base` sets the default base for HTML documents. A BASE tag within the HTML document overrides the command line setting. If the option `-base` is used the option `-html` can be omitted.

You can give the text certain attributes by entering them directly in the HTML file (see page 92).

VERSION CONTROL

`hwinstext` uses the options `-checkout`, `-commitversion` and `-forcedcheckin` to handle version controlled documents or cause new documents to be version controlled. See page 93.

EXAMPLE

```
hwinstext -hwhost intranet.mycompany.com -parent testcoll -html
-file 'example1.html example2.html'
```

This inserts two files which contain HTML document(s) into collection "testcoll" on the server `intranet.mycompany.com`.

4.6 VERSION CONTROL OPTIONS

`hwupload`, `hwinsdoc` and `hwinstext` allow you to use the three options `-checkout`, `-commitversion` and `-forcedcheckin`. Below is a brief definition of each.

`-checkout`

Checks out documents if they are version controlled.

`-commitversion`

Commits (checks in) documents if they are version controlled.

`-forcedcheckin`

Checks in documents which were not previously version controlled

These options have different effects depending on the combination in which they are used and whether or not documents are being replaced or newly uploaded.

Here we look at the eight cases there are for using these options when **uploading a new document**:

1. No options.
2. `-checkout`
3. `-commitversion`
4. `-forcedcheckin`
5. `-commitversion -forcedcheckin`

6. –`commitversion -checkout`
7. `-forcedcheckin -checkout`
8. `-commitversion -forcedcheckin -checkout`

The results when these options are used:

1. Document inserted; not version controlled.
2. Same as 1.
3. Same as 1.
4. Document inserted, checked in and checked out again.
5. Document inserted and checked in.
6. Same as 1.
7. Same as 4.
8. Same as 5.

When **replacing a document** there are many different cases depending on if the document to be replaced is version controlled or not and, in the case that it is version controlled, if it is checked in or checked out. All of these cases are described below.

1. No options

All cases: Document replaced; not version controlled

2. `-checkout`

Version controlled and document checked in: Document checked out and experimental version replaced

Version controlled and document checked out: Experimental version is replaced

Not version controlled: Document is replaced

3. `-commitversion`

Version controlled and document checked in: Error; document not replaced.

Version controlled and document checked out: Experimental version is replaced then committed.

Not version controlled: Document is replaced

4. `-forcedcheckin`

Version controlled and document checked in: Error, document not replaced.

Version controlled and document checked out: Experimental version is replaced

Not version controlled: Document is checked in and then out and is then replaced.

5. `-commitversion -forcedcheckin`

Version controlled and document checked in: Error, document not replaced.

Version controlled and document checked out: Experimental version is replaced then committed.

Not version controlled: Document is checked in and then out, replaced and committed.

6. –`commitversion -checkout`

Version controlled and document checked in: Document checked out, replaced, committed.

Version controlled and document checked out: Experimental version is replaced then committed.

Not version controlled: Document is replaced.

7. `-forcedcheckin -checkout`

Version controlled and document checked in: Document checked out, replaced.

Version controlled and document checked out: Experimental version is replaced.

Not version controlled: Document is checked in, checked out and replaced.

8. `-commitversion -forcedcheckin -checkout`

Version controlled and document checked in: Document checked out, replaced, committed.

Version controlled and document checked out: Experimental version is replaced then committed.

Not version controlled: Document is checked in, checked out, replaced and checked in.

4.7 VIEWING AND EDITING ATTRIBUTES

There are two Hyperwave tools which have functions that have to do with the attributes of Hyperwave objects. `hwinfo` allows you to retrieve attribute information about objects and `hwmodify` lets you edit attributes.

4.7.1 HWINFO

`hwinfo` is used to search in the Hyperwave database and to get information about objects. It allows you to search for any attribute (for example **Author**, **TimeCreated**, **DocumentType**, **Rights**, **Title**, **Keyword**) and to get all attributes or only one specific attribute (option `-attribute`) or only the object ID (option `-ids`) of the matching objects.

RUNNING HWINFO

`hwinfo` is run by entering `hwinfo` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING AN INITIAL SET OF OBJECTS

You must select the object or group of objects for which you want attribute information. There are four ways to select an initial set of objects which can then be filtered. The object can be specified exactly with `-object` *object*, where *object* is the name or object id of the object. The children of a collection can be selected using `-collection` *collection* where *collection* is the name or object id of a collection. It is also possible to use a KeyQuery or a Full Text Query to select objects. A KeyQuery is a Hyperwave database query which is performed on indexed attributes. `-keyquery` *keyquery* specifies the KeyQuery term, and `-language` *lang* replaces the default language for the query on titles, overriding the HWLANGUAGE environment variable. The `-formatted` option is used to specify that the keyquery is already formatted. You can specify `-collection` *collection* with the name or id of a collection and all selected objects will be descendants of this collection. See page 109 for KeyQuery syntax. The Full Text Query (`-ftquery` *ftquery*) lets you search for documents according to their contents. See page 110 for syntax.

SELECTING ASSOCIATED OBJECTS

After the options for selecting an initial set of objects, you can use further options to additionally select other objects associated with this set. `-recursive` selects all descendants, that is, recursive children of selected collections. `-anchor` selects all the anchors of the selected documents and `-parents` selects the parents of the selected objects (only the parents are shown).

Lastly, the preselected objects can be filtered using a Hyperwave ObjectQuery (`-query` *oquery*), which, in contrast to the KeyQuery, allows you to search for any attribute with enhanced matching capabilities. See page 111 for syntax.

Further optional parameters for `hwinfo`, as well as the environment variables they override, are: `-sortorder` *sortorder* (HWSORTORDER) which sets the sort order for output of multiple objects (see next paragraph), `-multiple`, which displays multiple objects (note that multiple is always true if options `-recursive` and `-anchor` are used), and `-ids`, which prints only a list separated by spaces of object ids of the selected objects. The option `-attribute` *attribute* can be used if you want to see only the value of the specified attribute; and if you have multiple objects you have to use `-multiple` too.

SORT ORDER

The output can be sorted according to certain attributes. Your preferred sort order can be communicated to `hwinfo` using `-sortorder` *sortorder* (see above). The value of *sortorder* must be a sequence of one or more of the abbreviations in the following table:

Attribute	Abbreviation
Author	A
TimeCreated	C
TimeExpire	E
TimeOpen	O
Score	S
Type (order: collection, cluster, text, image, film, audio, scene, PostScript, generic, annotation, remote)	t
Title	T
Sequence number	#
use the reverse of the next stated sorting criterion	-

EXAMPLES

```
hwinfo -hwhost my.testserver.com -collection '~anonymous'
-multiple -sortorder '#T' -attribute Title
```

This retrieves the titles of all children of collection "~anonymous" on the server `my.testserver.com` sorted by sequence number first, and then by title

```
hwinfo -hwho my.testserver.com -c '~anonymous' -m -s '#T' -at
Title
```

This is the same as above but uses the possibility to abbreviate options.

```
hwinfo -hwhost intranet.mycompany.com -coll '~anonymous' -sort
'#T' -attr Title -rec
```

This gets the titles of all descendants of collection "~anonymous" because the `-recursive` and thus `-multiple` options are set.

```
hwinfo -hwhost my.intranetserver.com -form -key 'Author=fmaier'
-coll technical -anch
```

In this example all descendants of collection "technical" on the server `my.intranetserver.com` are searched for Author "fmaier". In addition to all documents which match the KeyQuery, all anchors of the selected documents are shown.

```
hwinfo -hwhost my.intranetserver.com -form -key 'Author=fmaier'
-coll technical -anch -rec
```

This is same as above but option `-recursive` is set. Note that if an object in the KeyQuery result is a collection, in addition all descendants of this collection are selected. Do not use `-recursive` with KeyQueries unless you really know what you are doing!

4.7.2 HWMODIFY

`hwmodify` is used to change the attributes of a selected set of objects.

RUNNING HWMODIFY

`hwmodify` is run by entering `hwmodify` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING AN INITIAL SET OF OBJECTS

You must select a set of objects for `hwmodify` to work with. This is done in a way similar to the object selection with `hwinfo` except that here there are only three options instead of four. You can use exact specification (`-object` *object*), select children of a collection (`-collection` *collection*) or enter a KeyQuery (`-keyquery` *keyquery*). These options are explained on page 95. There is no Full Text Query option with `hwmodify`.

Two other options, `-recursive` and `-anchor`, can be used to select other objects associated with the group of objects selected above. See page 95 for more information about these options.

You have the possibility to further filter the group of selected objects using an object query (`-query` *oquery*). See page 111 for object query syntax.

THE MODIFICATION COMMAND

The modification command (`-command` *command*) is used to add and remove attributes and is required when using `hwmodify`. It is applied to all selected objects. The command can be made up of a series of elementary commands. There are three types of commands.

- `rem` *attribute*: removes all instances of the specified attribute
- `rem` *attribute=value*: removes only the specified attribute which has the specified value
- `add` *attribute=value*: adds the specified attribute and gives it the specified value

Your command can consist of a single command or you can put several elementary commands together, separating them with a backslash. In this way, it is possible to modify several attributes at once.

A NOTE ABOUT DIFFERENT ATTRIBUTE TYPES

There are several different attributes which you can give to your Hyperwave objects. Some of the attributes, e.g. `Name`, `Keyword`, etc. can be given to objects multiple times. Others, such as `Author` and `Rights` can be given only once. The reason for this lies in the nature of the individual attribute. For example, several `Name` attributes can be given to an object so that it can be referenced with different URLs, but it makes little sense to give an object several different (and possibly conflicting) `Rights` attributes.

Also note that when removing attributes of types which can only occur once per object (e.g. `Rights`) it is not possible to remove only those fields that have a specified value. This means that if you use a modify command such as `'rem Rights=R:a'` on a group of objects, the `Rights` fields will be removed for all the selected objects, and not just for those where the value is "R:a".

Note: *You cannot change attributes which are maintained by the server such as* `TimeModified`.

Note: *You cannot change the attributes of version controlled objects which are checked in.*

FURTHER OPTIONS

Another optional parameter is `-list`, which displays only a list of selected objects and does not apply the modification command. The option `-multiple` must be used when modifying multiple objects.

EXAMPLES

```
hwmodify -hwhost my.intranet.org -o technical/doc -comm 'rem Keyword'
```

This removes all `Keyword` fields from the object with the name "technical/doc" on the server `my.intranet.org`.

```
hwmodify -hwhost intranet.iicm.edu -coll technical -rec -mult -comm 'rem Rights\add Rights=R:g iicm'
```

Changes the value of the `Rights` field of all objects under collection "technical" on the server `intranet.iicm.edu` to "R:g iicm".

```
hwmodify -hwhost host.xyz.com -form -key 'Name=xyz' -comm 'add Name=abc'
```

Adds an additional unique name to object with name "xyz" on the server `host.xyz.com`.

4.8 DELETING OBJECTS

There are two Hyperwave tools which are used for deleting objects: `hwdelobj` and `hwdelete`.

4.8.1 HWDELOBJ

`hwdelobj` is used to delete individual objects, children of a given collection, or any object which matches a given query.

RUNNING HWDELOBJ

`hwdelobj` is run by entering `hwdelobj` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING AN INITIAL SET OF OBJECTS

You must select a set of objects for `hwdelobj` to work with. This is done in a way similar to the object selection with `hwinfo` (see page 95). You can use exact specification (`-object` *object*, where *object* is a name or object id), select children of a collection (`-collection` *collection*, where *collection* is the name or object id of a collection) or enter a KeyQuery (with all options except `-collection` *collection*) with `-keyquery` *keyquery*. See page 108 for KeyQuery syntax. There is no full text query with `hwdelobj`.

In the next step you can use the option `-anchor` to additionally select all the anchors of the selected documents

The last step is to (optionally) filter the selected objects using a Hyperwave ObjectQuery with the option `-query` *oquery*. Object Query syntax is found on page 111.

Further optional parameters are `-multiple`, which is required to delete multiple objects, and `-confirm` which causes `hwdelobj` to request confirmation before deleting objects.

EXAMPLES

```
hwdelobj -keyquery graz* -multiple -confirm
```

Deletes all objects on the server which fulfill the query

```
Name=graz*||Title=en:graz*||Keyword=graz*
```

with confirmation.

```
hwdelobj -collection projects -anchor -query 'Rights=R:g iicm' -multiple
```

Deletes all children of collection "projects" (including anchors) whose **Rights** attribute is "R:g iicm".

4.8.2 HWDELETE

`hwdelete` can delete or unlink one or more objects from a Hyperwave Information Server and is used recursively. When an object is deleted, it is physically deleted from the server. In contrast, unlinking means that an object is only removed from a specified collection, but continues to exist in other collections. However, if an object is unlinked from the only collection it is contained in, then unlinking is the same as deletion.

RUNNING HWDELETE

The `hwdelete` tool is run by entering `hwdelete` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING THE OBJECTS TO DELETE OR UNLINK

You must select a set of objects for `hwdelete` to delete or unlink. The option `-object` *object*, where *object* is the name or object id of the object (collection or document) you want to delete or unlink, is required. If the object is specified without the `–parent` *parent* option, then the object is physically deleted from the server. If the `-parent` *parent* option is given, then *object* is unlinked only from the collection *parent*. The objects selected can optionally be filtered using an Object Query (`-query` *oquery*). See page 111 for syntax.

There are some further optional parameters. `-multiple` is used if you want to delete a collection even if it has more than one parent and `-noconfirm` specifies that you do not want `hwdelete` to ask for confirmation. `-dellinks` deletes all links associated with the selected objects.

EXAMPLES

```
hwdelete -object testcoll -multiple -dellinks
```

Deletes collection "testcoll" and its members recursively, including links, even if it is member of more than one collection.

```
hwdelete -object project4 -parent hw_projects -query Rights=R:a
-noconfirm
```

For the instance of the collection "project4" which is contained in collection "hw_projects", those objects are deleted which have "R:a" as value of the **Rights** attribute, without confirmation.

4.9 MOVING AND LINKING OBJECTS

Hyperwave comes with one tool which can be used to move and link objects on the server.

4.9.1 HWMVLN

`hwmvln` can be used to move, link or unlink collections and documents in Hyperwave. As you may already know, all documents in Hyperwave are included in collections which are organized in a treelike structure. `hwmvln` is used to manipulate the tree structure by moving or linking collections or documents to different locations or unlinking objects from collections. Note that a link is not a physical copy separate from the original, but rather only a link to the same object. Unlinking is removing an object from a specific collection and not from the server.

RUNNING HWMVLN

`hwmvln` is run by entering `hwmvln` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING AN INITIAL SET OF OBJECTS

As with many of the other tools, you must select a set of objects for `hwmvln` to work with. There are three options for selecting objects: exact specification (`-object` *object*), select children of a collection (`-collection` *collection*) or KeyQuery (`-keyquery` *keyquery*). See page 95 for an explanation of these options.

You have the possibility to further filter the group of selected objects using an object query (`-query` *oquery*). See page 111 for syntax.

Below are listed the commands for moving, linking and unlinking collections and documents. In all cases, collections are specified using names or global object ids. You must use one of these options.

- `-link` *collection* links the selected objects to the specified collection.
- `-unlink` *collection* unlinks the selected objects from the specified collection.
- `-move` *source_coll destination_coll* moves the selected objects from the source collection to the destination collection.

Further optional parameters are `-multiple`, which is required when linking, unlinking or moving more than one object, and `-list`, which does not link, move or unlink, but only displays a list of the objects selected.

Note: If you want to link, unlink or move more than one object you have to use the option `-multiple`.

EXAMPLES

```
hwmvln -key vrweb -mult -link '~vrweb_user'
```

Links all accessible objects on the server which have "vrweb" in the English title or as keyword to the collection with the name "~vrweb_user".

Note: *The selected objects are not copied but rather just linked to the collection "~vrweb_user".*

```
hwmvln -coll '~vrweb_user' -mult -unlink '~vwreb_user'
```

Unlinks all objects from collection "~vrweb_user".

Note: *Even if you do not have write access to these objects and thus are not allowed to delete them, you can still use the command with these options to unlink all documents.*

```
hwmvln -coll Archives -mult -query 'TimeExpire<97/01/01' -move
Archives Archives.Erase
```

Moves all objects in collection "Archives" which expired before January 1, 1997 to collection "Archives.Erase".

Note: *Moving an object from collection "src" to collection "dst" can also be performed by linking it to collection "dst" and subsequently unlinking it from collection "src".*

4.10 COPYING OBJECTS

`hwcopy` is used for making physical copies of documents on the server.

4.10.1 HWCOPY

Unlike `hwmvln`, which only makes a link to a document, `hwcopy` makes an actual physical copy of the document on the server. Note that, at this time, only individual documents and not entire collections can be copied.

RUNNING HWCOPY

`hwcopy` is run by entering `hwcopy` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool. Default host is localhost, default port is 418.

REQUIRED OPTIONS

With `hwcopy`, the document to be copied must be given in the command line with `-object` *object*. The value of *object* can be the name or object id of the document. The destination collection for the copy is given with `-parent` *parent*, where the value of *parent* is the name or object id of the collection.

EXAMPLE

```
hwcopy -identify -object results/1997 -parent olddocs
```

Makes a copy of the document with the **Name** "results/1997" in the collection "olddocs".

4.11 TOOLS FOR VERSION CONTROL AND LOCKING OBJECTS

4.11.1 HWCI

The `hwci` tool is used to check in one or more Hyperwave documents. By default it checks in only documents which are version controlled and currently checked out. It does not check in documents which are not already version controlled unless the option `-forcedcheckin` is used (see below). If a version controlled document is already checked in, `hwci` does nothing with it unless the `-number` option is used (see below).

RUNNING HWCI

`hwci` is run by entering `hwci` *options* in the command line. The options of the tool are described below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING AN INITIAL SET OF OBJECTS

The documents to be checked in are selected using the options `-object`, `-collection`, `-keyquery`, `-language` and `-formatted`. See page 95 for more information about these options.

OBJECT QUERY

You have the possibility to further filter the group of selected objects using an object query (`-query` *oquery*). See page 111 for object query syntax.

RECURSIVE CHECK IN

By default, `hwci` only considers documents found in the selection process above and ignores collections. If you use the option `-recursive`, collections which are found are processed by `hwci` recursively, i.e. every document found in the collection is checked in.

FURTHER OPTIONS

The option `-number` *number* can be used to give all the objects being checked in the version number *number*. If this is not possible (e.g. the current version number of a document is greater than or equal to the number) `hwci` outputs an error message and does not check in the

document. If a version controlled document is checked in and the `-number` option is used, then if the version number of the document is lower than the number stated with the option, the document is checked out, given the new version number and checked in. If the `-number` option is not used, the version numbers of the objects are raised by one in the minor version number and the major version number stays the same.

The option `-comment` *comment* is used to give all checked in objects the same comment.

In the default case, `hwci` only checks in documents which are already version controlled. If you use the option `-forcedcheckin`, documents which are not yet under version control are also checked in.

The option `-ignoreerrors` tells `hwci` not to stop after finding an error.

EXAMPLE

```
hwci -identify -collection papers -keyquery -formatted
'Author=jjones' -recursive -number 2.0 -forcedcheckin
```

Checks in recursive children of collection "papers" on localhost if the author attribute of the document is "hwsystem". Documents which were not previously version controlled are also checked in because of the `-forcedcheckin` option. All checked in documents are given the version number 2.0 unless the document already has a higher version number, in which case `hwci` outputs an error.

4.11.2 HWCO

The `hwco` tool is used to check out one or more Hyperwave documents. By default it checks out only version controlled documents which are checked in. The tool ignores non-version controlled documents unless the option `-forcedcheckout` is used (see below).

RUNNING HWCO

`hwco` is run by entering `hwco` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` ['*user* [*password*]'] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

SELECTING AN INITIAL SET OF OBJECTS

`hwco` uses the same parameters for selecting an initial set of objects as `hwci` does. They are explained above.

OBJECT QUERY

As with `hwci`, it is possible to use an object query with `hwco` in order to further filter the initially selected objects (see above).

RECURSIVE CHECK OUT

By default, `hwco` only considers documents found in the selection process above and ignores collections. If you use the option `-recursive`, collections which are found are processed by `hwco` recursively, i.e. every document found in the collection and its subcollections is checked out.

FURTHER OPTIONS

The option `-forcedcheckout` tells `hwco` to check in non-version controlled documents and then check them out again (by default non-version controlled documents are ignored).

The option `-ignoreerrors` tells `hwco` not to stop after finding an error.

EXAMPLE

```
hwco -identify -collection home -recursive -forcedcheckout
```

Checks in recursive descendants of collection "home" on localhost. Non version controlled documents are checked in and checked out again due to the `-forcedcheckout` option.

4.11.3 HWREVERT

`hwrevert` is used to revert to a specified version for one or more version controlled documents.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

RUNNING HWREVERT

`hwrevert` is run by entering `hwrevert` *options* in the command line. Its options are explained below.

SELECTING AN INITIAL SET OF OBJECTS

`hwrevert` uses the same parameters for selecting an initial set of objects as `hwci` does (see page 101).

OBJECT QUERY

As with `hwci`, it is possible to use an object query with `hwrevert` in order to further filter the initially selected objects (see page 101).

RECURSIVE REVERT

By default, `hwrevert` only considers documents found in the selection process above and ignores collections. If you use the option `-recursive`, collections which are found are processed by `hwrevert` recursively, i.e. every document found in the collection and its subcollections is reverted.

FURTHER PARAMETERS

The parameter `-number` *number* tells `hwrevert` the version number to revert to. If the version number does not exist, `hwrevert` outputs an error. This parameter is required.

The option `-ignoreerrors` tells `hwrevert` not to stop after finding an error.

EXAMPLE

```
hwrevert -hwhost www.server.com -identify -recursive -
collection '~archive' -number 2.1
```

This command reverts to version 2.1 for all version controlled documents contained recursively in the collection "~archive" on the host `www.server.com`. If a document does not have a version 2.1, `hwrevert` outputs an error.

4.11.4 HWDOCHISTORY

`hwdochistory` lets you get a listing of the version history of an object.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user* [*password*]`'`] and `-user` *user* function the same way as for the `hwupload` tool (see page 82).

RUNNING HWDOCHISTORY

`hwdochistory` is run by entering `hwdochistory` *options* in the command line. Its options are explained below.

SELECTING AN OBJECT

The object whose document history you want to retrieve is specified with the `-object` *object* parameter. The value of *object* is the **Name** attribute or object id of the desired object.

EXAMPLE

```
hwdochistory -identify -object report97
```

This command outputs the document version history for the object with the name "report97" on localhost.

4.11.5 HWLOCK AND HWUNLOCK

`hwlock` and `hwunlock` are tools used to lock and unlock documents respectively. They use the same parameters as `hwdochistory` (see above).

EXAMPLE FOR HWLOCK

```
hwlock -identify andy -hwhost www.myserver.com -object
company/confidential/results97
```

Locks the object "company/confidential/results97" on server `www.myserver.com` for the user "andy".

EXAMPLE FOR HWUNLOCK

```
hwunlock -identify hwsystem -object doc/minutes/meetings/july
```

Unlocks the object "doc/minutes/meetings/july" on localhost.

4.12 MIRRORING INDIVIDUAL COLLECTIONS

Hyperwave has tools which can be used to store a particular collection and its contents in a file and then re-insert the collection into the same server or insert it into another server. This is useful for making a backup of a specific collection or for mirroring a collection to another server.

The tools are called `hifexport` and `hifimport`. The prefix "hif" comes from "Hyperwave Interchange Format", which is the special format in which Hyperwave collections are stored. HIF describes all aspects of a Hyperwave collection tree, including the collection relations and links between documents, in a location-independent fashion. It also includes links to documents outside the hierarchy (but not the documents themselves). The resulting HIF file may be edited, postprocessed and uploaded to a Hyperwave Information Server (using `hifimport`).

4.12.1 HIFEXPORT

`hifexport` is a command line tool which is used by entering a line of the form `hifexport` *options object(s)* after the command prompt. The object(s) are the collection(s) or document(s) which you want `hifexport` to download. How to specify the objects is explained below. The various options allow you to state which server you want to download from, which type of object you want to download, etc., and are also explained below.

Some of the options can be shortened, which is shown below with square brackets, e.g. you can enter `-identify` or `-iden`. The options which must be followed by additional information are shown with a word in italics after them, e.g. `-hgho[st]` *name*.

`-h` or `-help` displays options and defaults of the `hifexport` tool.

`-hgho[st]` *name* specifies the server to download from (overrides environment variable HGHOST, default "hyperg"). *name* can be the name of the server, e.g. `www.hyperwave.com`, or its IP address, e.g. `129.27.153.8`.

`-hgpo[rt]` *port* is the port the server is listening to and is 418 by default. (overrides environment variable HGPORT.

`-iden[tify]` causes `hifexport` to prompt you for your Hyperwave user name and password. This option is required if you are downloading objects with restricted access rights.

`-loca[l]` restricts export to the local server.

`-text[only]` specifies not to download non-text documents.

`-spli[tdoc]` causes documents to be saved in individual files.

`-path` *path* specifies where to save the individual documents; implies `-split` (overrides environment variable HGPATH, default ".")

`-query` *query* exports only objects conforming to *query*. See section "Syntax for object query" for syntax of the query.

`-stri[pprefix]` *prefix* strips the specified prefix from all object names.

`-tagm[ode]` causes text and anchors to be merged and put in HIF.

`-pipe[try]` number of tries for pipedocument (default 10). In normal use this should not be touched!

SPECIFYING OBJECT(S) WHEN USING HIFEXPORT

The object(s) are the root(s) of the tree(s) or document which you want to download. Objects can be specified using a unique identifier such as a name (which all collections are required to have and which other objects can optionally have) or an object ID, which all objects have. This information can be found by looking up the attributes of the object using the WaveMaster (Hyperwave's WWW gateway). It is also possible to retrieve attribute information using the command line tool `hwinfo`.

MORE ON HIFEXPORT OPTIONS

By default the program downloads to standard output in one big stream. The stream is formatted such that it contains only printable 7-bit characters, and that the collection hierarchy and link structure can be extracted from it.

If you want to store the documents each in a separate file, use a combination of `-split` and optionally `-path`. The output of the corresponding objects will still go to standard output.

If you use `-local`, only objects which are on the server from which you export will be exported. The other objects (collections) that are remote, will be exported as remote collections and therefore will be remote collections if you import them to another server.

If you have CGI objects on your server, you will get a warning if you have to export any CGIs manually, since CGI objects can be platform dependent and security malformed.

By default all documents are uuencoded. The anchors of the documents are written to standard output separately. Note that this also applies to text documents; if you don't want to have text uuencoded and want the anchors merged into its body, use the option `-tagmode`. If you want to download text documents only, use the option `-textonly`.

To strip a common prefix from the collection or document names which may have been prepended to make the names unique (e.g. with `hifimport`), specify that prefix along with the option `-strippprefix`.

If you specify a query, note that collections which do not conform to your query are not searched for members that conform to it. For example, to export only objects that were created after January 28, 1997, even if they are in collections that are older, specify `-query 'DocumentType=collection||TimeCreated>97/01/28'`. Note that queries should always be escaped with quotes.

RETURN CODES

`hifexport` returns zero on success, non-zero otherwise.

EXAMPLES

```
hifexport -hghost www.hyperwave.com graz > graz.hif
```

With this command, you would download the collection with the name "graz" as well as all its contents from the IICM information server `www.iicm.edu`. Because this collection has no restrictions on its read access rights, anyone can download it and the `-identify` option is not necessary. Because the output of `hifexport` goes to standard output by default, you have to redirect it if you want to put it in a file. In this case, output is being redirected to a file `graz.hif`.

```
hifexport -iden -query 'Rights=R:a' smith/projects > smith.hif
```

With this command, you would download a collection with the name "smith/projects" from the server which is set with your HGHOST environment variable. You would, of course, only download those documents contained in the collection which fit to the query "`Rights=R:a`", i.e. those documents which have access rights which are restricted to the owner of the document. To be allowed to do this, you would need to identify yourself (and have system rights), and thus

you must use the `-iden` option. The output in HIF format would be put into the file `smith.hif`.

4.12.2 HIFIMPORT

`hifimport` is a command line tool used for importing collections which are in the form of HIF (Hyperwave Interchange Format) files into Hyperwave. It can be used by entering a line of the form `hifimport` *options collection [HIF file]* after the command prompt. The various options allow you to state which server you want to upload to, specify what to do if an object being imported already exists on the server, etc. These options are explained below.

`-h`, `-help` displays options and defaults of the `hifimport` tool.

`-hgho[st]` *name* specifies the host name or address of the server to upload to (overrides environment variable HGHOST, default is "hyperg" in your domain).

`-hgpo[rt]` *port* is the port the server is listening to. This overrides environment variable HGPORT, default 418.

`-auth[or]` *author*|`HIF` lets you partially change the default behavior of `hifimport`. By default, the objects are inserted under the ownership of the person using the command. Users with Hyperwave system privileges may change this behavior using `-author`. Specify `HIF` to use the object authors which are specified in the HIF file, or any user name to set ownership to this user.

`-auto[identify]` is used to try to auto-identify user (non-interactively). Auto-identification must be enabled by your Hyperwave account (see section on user accounts and groups).

`-user` is used to enter the user name when not using `-auto` or not wanting prompting

`-password` is used to enter password of user when not using `-auto` or not wanting prompting.

`-path` *path* tells `hifimport` where to search for files (HIF files that have been exported using `hifexport`'s `-splitdoc` option). This overrides environment variable HGPATH and default is the current directory.

`-pref[ix]` *prefix* prepends *prefix* to all collection names. Choose a unique prefix to guarantee uniqueness; otherwise `hifimport` may overwrite existing objects.

`-repl[icate]` informs the server that the objects inserted should be treated as replicas (copies) of the original objects. They then behave like "permanently cached" objects, i.e. whenever the original (remote) object is accessed, the replica (local copy) is served. Please note that you are not informed when the original is modified, so use this option for datasets that change infrequently or never.

`-mirr[or]`, if set, causes objects and anchors to be deleted on the import server if they are not in the HIF file.

`-over[write]` *status* specifies how copies (objects with the same title and same document type in the same collection) should be treated: Four values of *status* are possible (default is `never`):

`never`

Never overwrite existing objects. The object found in the server is kept as it is.

`old`

Replace "old" objects. When both the object in the HIF file and the object on the server have specified modification dates, these are compared and if the HIF object is newer, it replaces the database object. Otherwise, the creation date is checked.

This option also tries to recognize "damaged" objects, i.e. documents that have meta-information but no data attached. It is assumed that a previous run of `hifimport`

terminated while inserting this document (e.g., by having lost the connection to the server). One can just start `hifimport` again with the `-overwrite old` option to correct the problem.

`replace`

Always replace existing objects. The object found in the server is deleted and the object in the HIF file is inserted instead.

`insert`

Does not check for existing objects when inserting, i.e. copy them if they occur. Use this method if you are sure that no copies exist, since it is the fastest.

`-loca[l]` specifies to insert no source anchors to remote links will be inserted.

`-time[out]` sends timeout for documents in seconds. Default is 60.

collection is the name or id of the collection to insert the contents of the HIF file under.

HIFfile is the HIF file to import (default stdin).

MORE ON HIFIMPORT OPTIONS

By default this program imports from standard input in one big file. If you haven't exported in one file, but in one file per document ("one-file-each" mode), and you don't have the files in the current directory, you must specify the path with `-path` (or set the environment HGPATH).

If you wish to insert a HIF file through standard input, you will have to use auto-identification mode (unless the HIF file is modified to contain user name and password at the very beginning).

If you use option `-over` with `old` or `replace`, objects which are to be replaced have to be found. It will be checked in the following order, until one object is left or an error is announced: Repl, Names, Titles, Parents.

If an object is replaced, all attributes are replaced but the GOid (global object id) remains the same.

If you use the `-mirror` option, all objects should be matchable, which can easily be done by using the `-repl` option or use names for all objects.

Warning: *If you use* `-mirror`*, data will be deleted on the import server, so be sure that there is no collection that only exists in the collection you are mirroring, otherwise the collection will not only be unlinked but deleted.*

RETURN CODES

`hifimport` returns zero on success, non-zero otherwise.

EXAMPLES

```
hifimport -user vmayr -hghost www.icg.edu smith graz.hif
```

This would import the HIF file `graz.hif` into the collection named "smith" on the server `www.icg.edu`. The user inserting this collection would only have to enter her password after issuing this command because she already entered her user name with the option `-user`.

```
hifimport -auto -author HIF -over replace whats_new news.hif
```

This imports the HIF file `news.hif` to the server which is specified with the environment variable HGHOST into the collection "whats_new". The user tries to automatically identify, which can only be done if the attributes of his account are set properly. Because of the option `-author HIF`, the original author names in the documents in the collection are kept (note that this can only be done by system users). Because of `-over replace`, any documents on the server that are also present in the HIF file will be replaced by the documents in the HIF file.

4.13 FETCHING DOCUMENTS

`hwgetdata` fetches document representations of one or more documents.

4.13.1 HWGETDATA

`hwgetdata` fetches document representations which are either written to standard output or stored each in a separate file. It is run by entering `hwgetdata` *options* in the command line. Its options are explained below.

The options `-help`, `-version`, `-hwhost` *hwhost*, `-hwport` *hwport*, `-identify` [`'`*user*[*password*]`'`] and `-user` *user* function the same way as for the `hwinscoll` tool (see above).

SELECTING AN INITIAL SET OF OBJECTS

As with many of the other tools, you must select a set of objects for `hwgetdata` to work with. There are three options for selecting objects: exact specification (`-object` *object*), select children of a collection (`-collection` *collection*) or Key Query (`-keyquery` *keyquery*). These options are explained on page 95.

Further options which can be used with the tool are `-recursive`, which additionally selects all descendants (recursive children) of the selected collections and `-anchor`, which adds anchors to text documents.

Preselected objects can be filtered using a Hyperwave ObjectQuery. This step is optional. The object query is specified with `-query` *oquery*. See page 111 for syntax.

Preselected items can also be filtered by document types using `-type` *type*.

A further optional parameter is `-multiple`, which fetches multiple document representations. Note that `-multiple` is always true if option `-recursive` is used. The option `-fpfx` *fpfx* changes the file name prefix. Without option `-fpfx` the file name has the form "<*loid*>.<*doctype*>", where *loid* is the local object id (e.g. "0x00006d47") and *doctype* corresponds either to the **MimeType** attribute (e.g. "text.html" for "MimeType=text/html") if it exists or to the **DocumentType** attribute (e.g. "Image" for "DocumentType=Image"). *fpfx* (including any relative or absolute path to an existing directory) is simply prepended to these file names, i.e. "<*fpfx*><*loid*>.<*doctype*>". If *fpfx* is equal to "-" (or for compatibility "stdout") the documents are written to standard output.

EXAMPLES

```
hwgetdata -obj technical/doc -fpfx -
```

This extracts the document with name "technical/doc" and writes it to stdout.

```
hwgetdata -coll tutorial -rec -fpfx 'tutorial.'
```

This extracts all documents from collection "tutorial" and prefixes filenames with "`tutorial.`".

4.14 QUERY SYNTAX

This section explains the syntax for the various query types which are used in combination with the tools.

4.14.1 KEY QUERY

A KeyQuery is a Hyperwave database query which is performed on indexed attributes. The following attributes are indexed:

- Author
- Keyword
- Name

- Title
- TimeCreated
- TimeModified
- Host
- Group
- UGroup
- UName
- UServer
- Repl
- Hint
- RemoteID

It is also possible to create and index arbitrary attributes in Hyperwave (see page 44). A KeyQuery can also be performed on these attributes.

RESTRICTING THE KEY QUERY TO A SPECIFIC COLLECTION

By default a KeyQuery is performed on the whole Hyperwave Information Server. With option `-collection` *collection* it is possible to restrict the KeyQuery result to the descendants of a given collection, which is specified using the name or object id of the collection. This option can also be used with simple and formatted KeyQueries.

SIMPLE KEY QUERY

A simple key query performs a key query on the attributes **Name**, **Keyword** and **Title**. The syntax is `-keyquery` *value*, where *value* is either a word, a word ending with an asterisk ('*') or a word ending with a '^'. Words ending with an asterisk find all words which have the word entered as a prefix, e.g. if you enter `austria*`, you would find objects which have "Austria", "Austrian" or "Austrians" as a name, keyword or part of the title. Words ending with '^' find all words with a value greater than or equal to the word entered. The option `-language` *lang* replaces the default language for the query on titles and overrides the environment variable HWLANGUAGE.

Examples

```
-keyquery graz*
```

The query `graz*` translates to the following key query term: `Name=graz*||Title=en:graz*||Keyword=graz*.`

```
-keyquery graz* -language ge
```

Because of the language option, this leads to the following key query term: `Name=graz*||Title=ge:graz*||Keyword=graz*`

FORMATTED KEYQUERY

Another option is to enter a formatted key query, which allows much more complex Hyperwave database queries. The Key Query term is specified with `-keyquery` *keyquery*, and `-formatted` is required to tell the tool that the key query is already formatted.

Formatted KeyQueries consist of attribute-value comparisons such as `Author=smith` or `TimeModified<97/03/03` which are combined with the AND (`&&`), OR (`||`) or ANDNOT (`&!`) operators, e.g. `Author=smith || TimeModified<97/03/03`.

Example:

```
-formatted -keyquery 'Title=en:Hyperwave&&Author=hwsystem'
```

This query is used to search the whole server for documents which have both the title "en:Hyperwave" and author "hwsystem".

KEY QUERY SYNTAX

Here is the exact syntax for KeyQueries:

```
keyquery    = "(" keyquery ")"
            | keyquery "||" keyquery                ; or
            | keyquery "&&" keyquery                ; and
            | keyquery "&!" keyquery                ; and not
            | keyfield "=" values
            | keyfield ">" simplevalue "<" simplevalue          ; range
```

```
keyfield    = "Title" | "Keyword" | "TimeCreated"
            | "Author" | "Name" | "UName"
            | "UGroup" | "Host" | "Group" | ...

values      = value
            | value whitespace values          ; same as "and"-operator

value       = simplevalue          ; normal word
            | simplevalue "*"      ; all words with prefix simplevalue
            | simplevalue "^"      ; all words greater or equal to simplevalue
```

A KeyQuery is a combination of one or more terms which give a desired attribute value or value range which are combined using various operators.

The query can consist of only a single word (see Simple Key Query above) or you can specify a desired attribute value or range with a formatted KeyQuery. Attribute types which can be used in the query are those which are listed on page 108. To search for objects with an attribute of a specific value, the syntax *attribute=value* is used, where *value* can be a word as described under Simple Key Query or it can be more than one of those words separated by spaces. When there is more than one word, they are ANDed together. Some possible queries are:

`Author=gmes*`

which finds all objects which have an author with a name that starts with `gmes`,

`Keyword=graz* styria`

which finds all objects which have both `styria` and a word with the prefix `graz` as keywords,

`TimeModified>96/03/01<97/01/15`

which finds all objects which were modified between March 1, 1996 and January 15, 1997.

These queries can be combined to more complex queries using the AND (&&), OR (||) and ANDNOT (&!) operators.

EXAMPLES

`TimeModified>96/03/01<97/01/15&&Author=ajancke`

This finds those objects which are owned by user "ajancke", and which were modified between March 1, 1996 and January 15, 1997.

`Author=gmesaric||Author=fkappe`

This finds documents which were inserted by `gmesaric` or `fkappe`.

4.14.2 FULLTEXT QUERY

A Fulltext Query is a query which is used to search in the content of documents. The formal syntax of a fulltext Boolean query is

```
expr        = term 1*(orop term)
term        = factor 1*(andop factor)
factor      = node ["{" float "}"]
node        = word 1*word  |  "(" expr")"
andop       = "&&" [ optionlist ]
orop        = "||"
optionlist  = "[" options "]"
```

options	= option 1*("," option)
word	= any word composed case- insensitively of validchars
option	= "F" | "f"
validchars	= any of abcdefghijklmnopqrstuvwxyzABC DEFGHIJKLMNOPQRSTUVWXYZ012345 6789-/

Examples:

```
hyperwave{6} || ftserver && joerg
```

This term would find documents containing the word "hyperwave" or both the words "ftserver" and "joerg", where documents containing "hyperwave" should be considered six times as important as documents containing the other ones.

```
hyperwave &&[f] ftserver
```

This term would find documents that contain both "hyperwave" and "ftserver", but the "&&" is not interpreted strictly because of the `f` (fuzzy) option. Rather a document containing only one of the two words is ranked far below one containing both words.

4.14.3 OBJECT QUERY

Object Queries are used to further filter out the results of a search that used a Key Query or Fulltext Query. While the Key Query only allows you to perform queries on indexed attributes, the Object Query lets you search for *any* attribute with enhanced matching capabilities.

Object Queries are constructed using the syntax *attribute operator value*, where *attribute* is any attribute, *operator* is "<", ">", "~" or "=" and value is the corresponding value. Some possibilities are:

```
-query 'DocumentType=collection'
```

which finds documents which are collections and

```
-query 'TimeExpire<97/01/01'
```

which finds all documents which expired before January 1, 1997.

These simple queries can be combined using AND (&&), OR (||), or NOT (!), as with KeyQuery. Queries should always be escaped with quotes.

Here is the formal syntax for Object Queries:

objquery	= "(" objquery ")"	
	| "!" objquery	; not
	| objquery "||" objquery	; or
	| objquery "&&" objquery	; and
	| attribute operator value	
attribute	= "Title" | "Rights" | ...	; any attribute name
operator	= "<"	; less than (strings)
	| ">"	; greater than (strings)
	| "~"	; regular expression

| "=" ; equal

Examples

```
-query '\!DocumentType=collection'
```

Finds documents which are not collections. Note that we have to escape `!` in the shell using `\!`.

```
-query 'TimeExpire<97/01/01'
```

Finds all documents which expired before January 1, 1997.

```
-query 'Rights~.*W:u[^;]*hwsystem'
```

Finds documents where part of the rights string is W:u hwsystem, using regular expression matching.

4.15 CGI AND HYPERWAVE

The Common Gateway Interface, or CGI, is the de-facto standard for interfacing external applications with information servers, such as Hyperwave or other Web servers. A CGI program is executed on the server in real-time, so that it can process input and generate dynamic information for output.

A detailed description and documentation for the Common Gateway Interface can be found at the "Hoohoo" server which is run by the NCSA HTTPd Development Team under URL

`http://hoohoo.ncsa.uiuc.edu/cgi/`.

CGI OBJECTS

In Hyperwave, CGI programs are not actually uploaded to the server, but rather they are put in a special directory meant for CGI programs. Hyperwave CGI objects must be created to reference the programs. These are objects of type "CGI" which contain the name and location of the actual CGI program in their **Path** attribute. These objects have to be called by other objects (e.g. HTML forms) to start the CGI program.

The **Path** attribute of a CGI object consists of exactly two parts. The first part is a directory name and the second part is the name of the CGI program, for instance:

```
Path=cgi-bin/myprog
```

if the program resides in the directory `cgi-bin` and is called `myprog` (see below for absolute location of the program).

Optionally the program name is followed by extra path information `/extra-path` and/or a query string `?query`. This can be used to provide a fixed query string (this information is added every time the CGI program is started). Optionally you can use the syntax: `<cgidir>/<cgi-name>?<query string>` for providing CGI access with dynamic parameters. Note that with the WWW gateway the CGI program is triggered using a URL, which can lead to ambiguities if you use the extra path option. The gateway resolves URLs by the Hyperwave **Name** attribute, so you have to name your CGI object if you intend to use it over the gateway. An excerpt from a typical CGI object's attributes would look like this:

```
...
Name=cgi-bin/cgi.pl
Path=cgi-bin/cgi.pl
...
```

If you reference this object with the following URL:

`http://<host>/cgi-bin/cgi.pl/my-extra-path/this/and/that`

then the gateway will look in the database for an object with the name `cgi-bin/cgi.pl/my-extra-path/this/and/that` which of course does not exist. To correct this problem, you have to tell the gateway explicitly where the CGI program name stops by writing the following in your configuration file `.db.contr.rc`:

```
WAVEMASTER::CGI_PATH = cgi-bin/
```

Note: Replace `WAVEMASTER` *with the name of your custom gateway if you have configured one.*

Each time a client requests a CGI object the server will execute the corresponding CGI program in real-time. The output of the program will go more or less directly to the client.

INSERTING CGI OBJECTS WITH WAVEMASTER

1. Identify as a system user.
2. Click on **Authoring** to switch to authoring mode.
3. Select **Publish→CGI**. The "New CGI" window appears.
4. Give the CGI object a title and select a language from the list.
5. Enter the **CGI-Path** for the CGI program, which consists of *directory/progname*.
6. Enter a **Name** attribute for the CGI object, and an optional description.
7. Click on "More" to enter additional attributes. The **Custom Attribute** field allows you to enter an attribute with the name and value of your choice.
8. Click on the "OK" button.

INSERTING CGI OBJECTS WITH HWINSDOC

See also page 90.

Insert your CGI object with command

`hwinsdoc -hwhost` *your_host* `-parent` *name_or_id* `-title` *title_required* `-name '`*name*`' -type C -relurl '`*directory/progname*`'`

replacing the parameters in italics with the corresponding information.

CGI PROGRAMS

In Hyperwave CGI programs reside outside the database in the special server directory `~hwsystem/dcserver/cgi`. This is the entry point for the server module `dcserver` which executes the CGI programs. The first two path components of the `Path` attribute in the corresponding CGI object are interpreted as a relative path name beginning in the `~hwsystem/dcserver/cgi` directory. For example, if the path attribute reads

```
/cgi-bin/myprog
```

WHERE TO PUT THE CGI PROGRAM

the server assumes the executable is physically present at `~hwsystem/dcserver/cgi/cgi-bin/myprog`. With this scheme (restricting the number of path components to exactly **two**) ambiguities between the path to the executable and additional path parameters as described above are prevented.

The Hyperwave Information Server provides a default directory `~hwsystem/dcserver/cgi/cgi-bin` to deposit your programs. However, you are encouraged to create any number of directories and/or links from `~hwsystem/dcserver/cgi`, as long as the relative path to the executable is exactly **two** components.

The `cgi` directory is normally under direct control of the server administrator (usually the user `hwsystem`), prohibiting other users, who have no write access to this directory, from creating CGI programs.

Note: On the Windows NT server, Perl CGI scripts must have the extension `.pl`.

FORMS

A CGI object is usually called from within a `<FORM>` tag. This is done by a keyword that you place in the ACTION field of the `<FORM>` tag. The keyword acts as a hint for the insert tool (`hginstext`) to look in the database for a CGI object that has a **name** which is the same as the keyword. If such an object exists, a link is created immediately. Otherwise an open link is placed into the database, which will be resolved when an object with the appropriate name comes along

later on. When the form is submitted, the link is followed, which causes the CGI program to be executed with the form data.

Insert the form like every HTML text with your Web browser or `hwinstext`.

Although the use of CGI is strongly related to forms, a CGI program may not only be called by submitting the form data. Rather, it is also possible to execute it directly over its associated object.

COMPLETE CGI EXAMPLE

The following example will show you how to write a Perl script, how to insert it into the server, how to create the associated object and how to call it from an HTML form. The program will repeat a text inserted by a user and print out all CGI environment variables.

1. Write the following script, give it the name '`cgi-test.pl`' and place it under ~`hwsystem/dcserver/cgi/test/`. The script must be executable.

```
#!/usr/local/bin/perl
print "Content-type: text/html\n\n" ;
print "<HTML><HEAD><TITLE>CGI Test Script Output</TITLE>\n" ;
print "</HEAD><BODY>\n" ;
print "<PRE>\n" ;
print "The arguments are: ", join (' ', @ARGV), "\n\n" ;
print "The CGI environment is:\n\n" ;

print "SERVER_SOFTWARE = ", $ENV{"SERVER_SOFTWARE"}, "\n" ;
print "SERVER_NAME = ", $ENV{"SERVER_NAME"}, "\n" ;
print "GATEWAY_INTERFACE = ", $ENV{"GATEWAY_INTERFACE"}, "\n" ;
print "SERVER_PROTOCOL = ", $ENV{"SERVER_PROTOCOL"}, "\n" ;
print "SERVER_PORT = ", $ENV{"SERVER_PORT"}, "\n" ;
print "REQUEST_METHOD = ", $ENV{"REQUEST_METHOD"}, "\n" ;
print "HTTP_ACCEPT = ", $ENV{"HTTP_ACCEPT"}, "\n" ;
print "PATH_INFO = ", $ENV{"PATH_INFO"}, "\n" ;
print "PATH_TRANSLATED = ", $ENV{"PATH_TRANSLATED"}, "\n" ;
print "SCRIPT_NAME = ", $ENV{"SCRIPT_NAME"}, "\n" ;
print "QUERY_STRING = ", $ENV{"QUERY_STRING"}, "\n" ;
print "REMOTE_HOST = ", $ENV{"REMOTE_HOST"}, "\n" ;
print "REMOTE_ADDR = ", $ENV{"REMOTE_ADDR"}, "\n" ;
print "REMOTE_USER = ", $ENV{"REMOTE_USER"}, "\n" ;
print "CONTENT_TYPE = ", $ENV{"CONTENT_TYPE"}, "\n" ;
print "CONTENT_LENGTH = ", $ENV{"CONTENT_LENGTH"}, "\n\n\n" ;

print "The stdin input is: \n\n" ;
print while <STDIN> ;

print "</PRE>\n" ;
print "</BODY></HTML>\n" ;
```

2. Create a CGI object which refers to the program with

```
hwinsdoc       -hwhost yourhost
       -parent name_or_id
       -title 'CGI test object'
       -name 'cgi-test-program'
       -type C
       -relurl 'test/cgi-test.pl'
```

3. Write an HTML text which calls the CGI object.

```
<HTML>
<HEAD>
<TITLE>A Simple CGI Test Form</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="cgi-test-program">
```

```
<EM>Insert some text here. It will then be echoed by the script.</EM><P>
<TEXTAREA NAME="text" ROWS=2 COLS=60></TEXTAREA><P>
<INPUT TYPE="submit" VALUE="Execute the script"></FORM>
```

4. Insert the form with

```
hwinstext      -hwhost yourhost
       -parent name_or_id
       -name 'cgi-test/form.html'
       -html
       -file formtest.html
```

5. Start your Web Browser and look at

```
http://yourhost/cgi-test/form.html
```

6. The output of your script should look like:

```
Test


The CGI environment is:

SERVER_SOFTWARE = Hyperwave Document Server (dcserver)
SERVER_NAME = www.hyperwave.com
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = POST
HTTP_ACCEPT =
HTTP_USER_AGENT = Mozilla/2.0 (compatible; MSIE 3.0; Windows 95) via
Squid Cache version 1.0.20
HTTP_REFERER = http://www.hyperwave.com/cgi-test/form.html
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /test/cgi-test.pl
QUERY_STRING =
REMOTE_HOST = fpuexx01.tu-graz.ac.at
REMOTE_ADDR = 129.27.99.40
REMOTE_USER = hgsystem
REMOTE_IDENT =
AUTH_TYPE = Basic
CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 20
HTTP_COOKIE = sk=01293C6FB11E
SESSION_KEY =
The stdin input is:

text=this+is+my+text
```

4.16 JAVA APPLETS AND HYPERWAVE

Java is one of the most recent and most exciting developments in the World Wide Web. It promises to solve many problems by being platform independent and offering new perspectives on interaction and information presentation over the Internet.

Java as well as JavaScript are supported by Hyperwave.

4.16.1 INSERTING JAVA CLASSES

Java classes must have the **DocumentType** "Program" and a **Name** attribute to be referenced from within an HTML document. JAVA classes can be inserted in two ways:

USING A WEB BROWSER

1. Press the **Edit** button on top of your screen (in default layout).
2. From the options that appear choose **Insert Document**.
3. In the form that appears you can insert required and recommended information, like **Collection** (name of the collection where you want to insert your applet), **Title**, **Name**, **File** (in some browsers you can browse for file names). From the listbox choose the attribute **DocumentType** and type in `Program` as value. The **Name** attribute consists of codebase/classname, for the example below **Name** is `mydir/java/xyz.class`.
4. Commit your entries by clicking on **Insert Document**.

USING COMMAND LINE TOOLS

Insert your Java class with command

```
hwinsdoc -hwhost your_host -parent name_or_id -title title_required -
name "mydir/java/xyz.class" -type Program -path file
```

replacing the parameters in italics with the corresponding information (see also page 90).

RUNNING JAVA

To run JAVA a reference must be included in a HTML document following the pattern:

```
<applet codebase="/mydir/java" code="xyz.class">
<param ....>
```

4.17 HOW LINKS ARE HANDLED WHEN DOCUMENTS ARE REPLACED (HINTS)

In order to preserve link consistency, links are normally automatically deleted when an object is deleted from a Hyperwave Information Server. However it is possible, with the tool `hwdelete`, to tell the server not to delete associated links when deleting objects. You can then replace the object and have all the links that pointed to the old object point to the new object.

This is done using so-called *destination hints*. A hint is certain information about the object being deleted. This information is saved and when Hyperwave finds that a document with the same information has been inserted, this object becomes the destination for the open links. What information is saved as a hint depends on the attributes of the object. There are two possibilities:

- If the object has a name, then the name is used.
- If it doesn't have a name, the name of the parent collection is used as part of the hint. The other part is the alphabetically first keyword, and if it has no keywords, then the (alphabetically) first title.

The hyperlinks will be reconstructed if the title of any object in the collection (or its subcollections) is changed to match the destination hint or if such a title is added. It does not work if an object with the same title is copied or moved into the collection.

5 TROUBLESHOOTING

If you experience any difficulties during installation or running of a Hyperwave Information Server, have a look at this chapter. Here the most common errors and problems and how to solve them are described.

5.1 ANALYZING SERVER PROBLEMS

This is a short guide to what you should do first if you have any problems with your server. Please follow these steps to try to locate the problem. If you cannot find out what the actual problem is you should have a look at the list of error messages and problem descriptions below which should help you to solve the problem. If this does not help you, please refer to the Help section at the end of this chapter to find out where you can get more information.

To find out where a problem might be located it is important to know that the server software consists of several different server processes (for a detailed description of them see Chapter 1 of this guide). A script called `hwservercontrol` controls all parts of these modules. Generally, it does two things:

1. It starts all necessary servers.
2. It restarts parts of the server which die unexpectedly. The restarts are counted and the whole server is killed if the count reaches a maximum (default 10) to avoid looping in the case of a severe problem.

The `hwservercontrol` script first reads the main configuration file (`$HOME/.db.contr.rc`) to get information about the processes to start and their log file/error file location. It writes all activity (starting and restarting of processes) into its own log file (specified with the `MAIN::log` option of `.db.contr.rc`). The default location is `$HOME/log/server.log`.

This main log file is the first place to look if an error occurred, to see which of the server processes has died. After starting the server, the content of `server.log` should look something like this:

```
hwservercontrol: 97/03/21 10:24:48 start
hwservercontrol: 97/03/21 10:24:48 starting dbserver
hwservercontrol: 97/03/21 10:24:48 starting DCSERVER
hwservercontrol: 97/03/21 10:24:48 starting FTSERVER
hwservercontrol: 97/03/21 10:24:49 starting HGSERVER 418
hwservercontrol: 97/03/21 10:24:49 starting WAVEMASTER 80
```

As you can see the script itself was started at 97/03/21 10:24:48. After that the `hwservercontrol` script started the following processes: `dbserver` (low level database server), `dcserver` (document cache server), `ftserver` (full text indexer), `hgserver` (link server which implements the HG-CSP), and `wavemaster` (the WWW gateway).

If one of these processes fails it terminates and the `hwservercontrol` script restarts it. If it's a severe problem this process will repeatedly terminate; and after ten restarts `hwservercontrol` stops the server and there is something like the following in `server.log`:

```
hwservercontrol: 96/03/21 14:57:45 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:46 starting WAVEMASTER 80
```

```
hwservercontrol: 96/03/21 14:57:46 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:46 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:47 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:47 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:47 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:47 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:47 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:47 starting WAVEMASTER 80
hwservercontrol: 96/03/21 14:57:48 starting WAVEMASTER 80
hwservercontrol: Error: too many loops
            WAVEMASTER      11
hwservercontrol: 96/03/21 14:57:48 killing (TERM, pg) child processes
hwservercontrol: 96/03/21 14:57:48 got signal TERM
hwservercontrol: 96/03/21 14:57:48 killing (TERM, pg) child processes
hwservercontrol: HGSERVER finished
hwservercontrol: FTSERVER finished
hwservercontrol: DCSERVER finished
hwservercontrol: dbserver finished
hwservercontrol: 96/03/21 14:58:03 killing (KILL, pg) child processes
```

You can see that the `wavemaster` process died and `hwservercontrol` killed all the other processes. From this you know that `wavemaster` is the problem and you should therefore have a look at the corresponding log files (`$HOME/log/wave.log` and `$HOME/log/wave-err.log`).

Please refer to page 6 about directory and log file organization to get more information on where the specific log files are located.

5.2 INSTALLATION AND CONFIGURATION PROBLEMS

5.2.1 LICENSE PROBLEMS

If you have any problems regarding licensing and registration, you should first try to find out whether you can solve the problem yourself by reading the problem descriptions below. If this is not possible, see the *Hyperwave Installation Guide* for what to do to obtain a valid license.

WRONG LICENSE FILE NAME

Problem: If the following line appears in `$HOME/log/server.log` then you might have given the wrong file name to your license file:

Error Message: `hwservercontrol: Error: [HGSERVER] FATAL: Error: Hyperwave-License $HOME/.server_license: Couldn't access file`

Solution: Make sure that the license file is called **.server_license** (note the American style with "s" in license) and that it can be read by the user `hwsystem`.

WRONG LICENSE FILE CONTENT

Problem: One of the following lines may appear in `$HOME/log/server.log` if the content of the license file is not correct:

Error Messages:

```
hwservercontrol: Error: [HGSERVER] FATAL: Error: Hyperwave-License: $HOME/.server_license Wrong license file
```

```
hwservercontrol: Error: HGSERVER FATAL: Error: Hyperwave-License: $HOME/.server_license Checksum error
```

Solution: Check the content of the `.server_license` file. The content of the file should not have been altered in any way. It has to start with 4 dashes ("`- - - -`") at the beginning of the BEGIN and of the END line and 5 dashes ("`- - - - -`") at the end of the BEGIN and of the END line, like the following:

```
- - - -BEGIN HYPERWAVE LICENSE- - - - -
- - - -END HYPERWAVE LICENSE- - - - -
```

After each line there must not be any additional characters or spaces.

WRONG LICENSE VERSION

Description: The following line appears in `$HOME/log/server.log` if you have a wrong license version. Note that licenses are only valid for major Hyperwave releases.

Error message: `hwservercontrol: Error: [HGSERVER] FATAL: Error: Hyperwave-License $HOME/.server_license: wrong license version`

Solution: Apply for a new license. In the meantime you may retrieve the test license from our server (`http://www.hyperwave.com/download/license`) to be able to work.

WRONG HOST ADDRESS

Description: The following line appears in `$HOME/server.log` if you have not specified the right IP address in your license application or are using the license file on the wrong machine.

Error message: `hwservercontrol: Error: HGSERVER FATAL: Error $HOME/.server_license: Hyperwave-License: wrong host address`

Solution: Check the IP address in the `Host:` field of your `.server_license` file. This should be the address which is used as the default network device. If the IP address of the server has changed apply for the changed license by contacting `support@hyperwave.com`.

LICENSE EXPIRED

Description: Unless you have purchased the Hyperwave Information Server software, you will have a limited license which expires after a certain period (`TimeExpire`). You will not be able to start the server after that date. Furthermore no new sessions will be accepted.

Error message: `hwservercontrol: Error: [HGSERVER] FATAL: Error: Hyperwave-License $HOME/.server_license: License expired`.

Solution: Apply for a new license at `http://www.hyperwave.com/download/license`.

5.2.2 HGBINDPORT PROBLEMS

HGBINDPORT NOT OWNED BY ROOT OR THE SETUID BIT NOT SET

Description: The Hyperwave Information Server communicates via the privileged port 418. The WWW gateway uses port 80 as default. Neither port can be used by processes belonging to normal users. To avoid running the whole server under userid 0 (root) a tiny program (`hgbindport`) is used to open this port. `hgbindport` then executes the real server process which inherits the open port. `hgbindport` must be **owned by root** and the **setuid bit** must be set.

Error message: `hwservercontrol: Error: starting HGSERVER: hgbindport(echo):bind(418): Permission denied`

Solution: Change the owner of `$HOME/bin/$CPU/hgbindport` to root and set the setuid bit (you have to have root permissions in order to do this).

HGBINDPORT CANNOT EXECUTE REAL SERVER

Description: If the server binaries are not on the server (that is if someone removed or moved them), `hgbindport` cannot find and execute them.

Error message (in the errorlog file of the appropriate server, for example `wavemaster`): `/home/hgsystem/bin/$CPU/hgbindport(wavemaster):execvp: No such file or directory`

Solution: Check which server fails (`$HOME/log/server.log`); change to the directory `$HOME/bin/$CPU` and look for the appropriate binary (in our example `wavemaster`).

PORT ALREADY IN USE

Description: On some systems (Linux in particular) an http daemon may be running as default. You need to remove it to be able to use port 80.

Error message: `hgbindport: port already in use`

Solution:

1. Find out what http daemon or WWW server is running and kill it. Make sure there is no such process started.
2. Run the server's WWW gateway on a different port. This can be configured using WaveSetup (see page 8).

5.2.3 PERL PROBLEMS

WRONG HEADER FILES

Description: To work with Perl you should have configured the Perl header files in the right way. Normally this is done by running h2ph, but some additional changes have to be made manually. If some values are missing, Hyperwave will take default values.

Error messages:

```
Perl warning: no sys/socket.ph, using default values instead
```

```
Number found where operator expected at (eval 174) line 1, near
") 0" (Missing operator before 0?)
```

```
Can't locate linux/socket.ph in @INC (did you run h2ph?) at
/usr/lib/perl5/i586-linux/sys/socket.ph line 4
```

Solution: If the server starts up without problems you can safely ignore these Perl messages. If you don't want to deal with configuring Perl header files, you may remove all header files. The default values will be taken.

5.2.4 HYPERWAVE INFORMATION SERVER BEHIND A FIREWALL

DISABLE SERVER-TO-SERVER-COMMUNICATION

Description: A Hyperwave Information Server which is running in an internal network (Intranet) only, should not communicate with the outside world. Depending on your license or the default settings your server may be configured to communicate with other servers. To avoid this behavior you can disable this feature.

Solution: Check with WaveSetup if Server Update is set to "yes" and switch to "no" if you want to stop your server from communicating with other servers.

UPLOAD DOCUMENTS THROUGH A FIREWALL

Description: A firewall normally allows only limited access from the outside world to any resources within the internal network and vice versa. To enable internal clients to access a Hyperwave Information Server which is located outside the firewall (or to enable external clients to access a server inside the firewall) certain ports have to be kept open.

Solution: Configure the firewall to open the port number 418 for exchanging commands with the server and port 4712 for uploading documents to the server.

5.2.5 WAVESETUP PROBLEMS

UNIX SYSTEMS WITH SHADOW PASSWORDS

Description: Under UNIX, by default the user name and password for WaveSetup are the name and password (read from the password field of `/etc/passwd`) from the account under which Hyperwave was installed. This doesn't work with shadow passwords.

Solution: In systems with shadow passwords, the user name for WaveSetup is taken from the account under which Hyperwave Information Server was installed and the password is "hwsystem". The password should be changed as soon as possible with WaveSetup (on the **General** page in the **WaveSetup** section) or by changing the variable

`MAIN::PASSWD =` *encrypted_password*

in Hyperwave's main configuration file `.db.contr.rc`.

Note: *The user name and password for the server and WaveSetup are completely independent, i.e. changing one does not change the other.*

CONNECTING TO WAVESETUP FROM A HOST OTHER THAN LOCALHOST

Description: After trying to log in to WaveSetup, the message

"Hyperwave WaveSetup: Access denied

WaveSetup does not allow administration from the hosts your browser is running on. Check that you are not connecting to WaveSetup via a proxy!"

appears.

Solution: For security reasons, WaveSetup allows by default only connections from the host the server is running on. If you want to access WaveSetup from a different host, this host must be configured with either the variable `WAVESETUP::ALLOWED_HOST` or `WAVESETUP::ALLOWED_IP` in `.db.contr.rc`. See also page 30.

STARTING WAVESETUP ON NT

Description: I have installed Hyperwave on NT following the description in the installation guide. As far as I can see, Hyperwave is active but I cannot use WaveSetup. When I start WaveSetup a DOS Window pops up but remains black. What is the problem?

Solution: For optimum convenience WaveSetup is started together with the server in the NT version. It is not necessary to start it manually. Connect to your server at port 9999 as described in the README file.

5.2.6 BACKUP PROBLEMS

PROBLEMS WITH BACKUP FOR NT

Problem: You are starting backup for the NT Hyperwave Information Server from the command line and it is not working.

Solution: Make sure that the `$HOME` environment variable is set properly, i.e. to the home directory of the Hyperwave Information Server.

5.2.7 PROBLEMS WITH STARTING AND STOPPING THE SERVER

NT SERVER IS STOPPED BUT SERVICES SAYS IT IS STILL RUNNING

Problem: Your NT Hyperwave Information Server is no longer responding but the Windows NT services window says the server is running.

Solution: When the NT Hyperwave Information Server stops due to an error, the Services window still says that the server is running even though it is not. However, the server configuration tool, WaveSetup, is running and can be accessed with the URL `http://`*<my.server.name>*`:9999`. The configuration tool can be used to make changes that will alleviate the error, e.g. if your server license has expired, you can enter the new license on the **General** page. If the problem is not with the license, check your server log files to find out what the problem might be.

NT SERVICE WINDOW ENTRY IS LOST

Problem: The entry used for starting the Hyperwave Information Server in the Windows NT Services window has disappeared, e.g. because of an error with setup.

Solution: Use the NTServiceControl tool to create the entry again. See the *Hyperwave Installation Guide* for instructions on how to do this.

OTHER PROBLEMS WITH STARTING THE SERVER

See also page 118, License Problems.

5.2.8 OTHER CONFIGURATION PROBLEMS

ARCHITECTURE NOT YET SUPPORTED

Description: During installation of Hyperwave Information Server on my UNIX machine I get the error message "architecture not yet supported".

Solution: To find out why your architecture doesn't work correctly with Hyperwave, please send us the output of the command

```
uname -a
```

A quick solution is to manually set the correct environment variable to one of the following (according to your system):

```
setenv CPU BSDI
setenv CPU LINUX_ELF
setenv CPU ALPHA_OSF1
setenv CPU HPUX9
setenv CPU IBMAIX
setenv CPU SGI
setenv CPU SUN5
setenv CPU SUN4
```

NO FILETABLE AVAILABLE (HWINSTTAR)

Description: While using `hwinsttar` to update the UNIX server, you get the error:

```
No filetable available
```

Solution: You probably started `hwinsttar` from within the `tartmp` directory. Change to the `hwsystem` directory and start the installation with `tartmp/hwinsttar`.

DBSERVER:NO SUCH FILE OR DIRECTORY

Description: You have installed Hyperwave Information Server for UNIX with the `hwinsttar` script, following the instructions from the *Hyperwave Installation Guide*, but cannot start the server with `hwstart`.

The `server.log` file contains something like the following:

```
dbserver.control: 97/05/07 20:06:08 start
dbserver.control: 97/05/07 20:06:08 starting dbserver
dbserver.control: Error: exec dbserver:No such file or
directory
...
dbserver.control: 97/05/07 20:06:24 killing (KILL, pg) child
processes
```

Solution: Check your `.cshrc` file. If there is a path setting statement, put the `source .hgrc` line after the last one of these statements. Your problem might be an exit statement (executed if shell is noninteractive) before reaching the `source .hgrc` line, which adds the Hyperwave directories to the path variable.

HW SERVER NOT RUNNING (WWW SERVER ON PORT 80)

Description: The Hyperwave Information Server cannot be started on my system.

Solution: If you already have an ordinary WWW server running on port 80 you must stop the other server or configure the server to use another port number.

LINUX ELF SHARED LIBRARIES

Description: To run a Hyperwave Information Server on a LINUX_ELF system you need certain shared libraries. Otherwise you will have problems with parts of the server.

Error message: `[ftserver] gets signal 11`

Solution: Check your libraries with:

```
ldd 'which ftserver'
```

You will need libc-5.4.13 or above, libg++-2.7.2.1, ld.so-1.8.1 or above, binutils-2.7.0.3 or above, gcc-2.7.2 or above

LOOPBACK CONFIGURATION

Description: The Hyperwave Information Server will not work on a loopback only system. You can either enable a "real" network interface or setup a "dummy" network device.

Solution: Please refer to your system documentation.

MULTIHOMING

Description: Multihoming can be configured with the WaveSetup tool. For each IP address (and hostname) a different entry point and a different user can be configured.

Solution: An example multihoming configuration may look like this:

```
WAVEMASTER::HOSTS = first.host.name 1.2.3.4 user1 entry1
WAVEMASTER::HOSTS = second.host.name 5.6.7.8 anonymous entry2
WAVEMASTER::HOSTS = * * anonymous /
```

***Note**: All objects on a server configured in the above way are still on **one** server. This means names have to be unique across this system, although it may seem to the user that these are different servers.*

ANONYMOUS USER CANNOT LOGIN

Description: When accessing a Hyperwave Information Server the WWW gateway (WaveMaster) tries to find an object with a name specified in `.db.contr.rc` (see the chapter about WaveSetup on how to configure this, or the section on multihoming above). If there is no such entry, it looks for an object with the name "/". If this does not exist, a collection called "rootcollection" will be accessed.

Error message: Object not found or Identification dialog appears.

Solution: Check if there is any such entry about multihoming or entry points. If this is not the case, search for an object called "/" If this does not exist, but you find an "~anonymous" or "rootcollection", then assign the name "/" to one of them, to make it the default entry point of your

server. Finally, if all names exist already, check the access rights and provide at least one anonymously accessible page to explain to the user how to proceed to access the server.

ROBOTS GENERATE MULTIPLE SESSIONS

Description: If so-called robots access a server, they should identify themselves as robots. If everything works correctly, each robot only gets one session. If robots create multiple sessions, this may cause trouble because they will try to access every object on the server and thereby create many sessions.

Solution: There is a file called `robots.idx` in your `$HOME/wavemaster/idx` directory (configurable with WaveSetup) which contains all known robots. If you add the intruding robot there this should solve the problem. With the following entry you can even let robot requests run with a different standard entry point, under a different account and thus with different access rights than standard user requests.

```
WAVEMASTER::DEFAULT_ROBOT = robot /
```

5.3 RUNTIME PROBLEMS

5.3.1 PROBLEMS WITH PARTS OF THE SERVER

FULLTEXT INDEX SEEMS TO BE INCOMPLETE

Description: The fulltext index may become incomplete if large amounts of data are inserted and `ftserver` crashes or gets killed while insertion is still going on (compare the logfiles to find out if this has happened). There are various actions that can be taken to help the `ftserver` recover from errors. There are several kinds of errors/inconsistencies that can occur:

1. Some documents have not been indexed for some reason.
2. You have changed the configuration, e.g., added some stop words or the like.
3. You have switched the engine to an engine you have used some time ago. The `ftserver` does not currently recognize that the indexes are outdated. Note that this reduces to case 2, because all engines share the same internal table of documents.

Solution:

1. Create a file named `BUILDREP` in the `$HOME/ftserver` directory. Kill the `ftserver`. The `ftserver` will then create the internal table of documents it has indexed/should index. After having done this, it will rebuild the indexes of the engine that is used.
2. Create a file named `REBUILD` in the `$HOME/ftserver` directory. Kill the `ftserver`. The `ftserver` will rebuild the indexes of the engine that is used.
3. See 2.

CONNECTION TO LOW-LEVEL DATABASE FAILED

Problem: When connecting to my Hyperwave Information Server with a web client I get the error message:

```
Error Hyperwave Server:Connection to low-level database failed
(HW_DBSTUBNG)
```

Solution: This may occur after the server has crashed and is reorganizing its data. During this time you cannot connect to the server. The duration of this process depends on the amount of data on your server and ranges from several minutes to over an hour.

5.3.2 PROBLEMS WITH TOOLS

ENTRY IN NAME FIELD NOT UNIQUE

Description: Name attributes are unique identifiers in Hyperwave. Thus each name can exist only once on the server. If you insert an object with the Hyperwave tools into the server and want to give it a name in one run, you must ensure that the names they are given are unique. With some tools names are taken from filenames. It must be guaranteed that there is no such object on the server already.

Error message: Entry in name field not unique.

Solution: Check if there is no object with a name that you want to pass over with your filename. If such an object already exists, rename it or upload your object with a prefix.. If you do not actually want to give the object a filename, check if there is a `<META Name="">`-tag in the HTML text and whether this name already exists in the database. If this is the case, remove the `<META>`-tag and try inserting the object again. (Some attributes, such as names may be transferred into Hyperwave by parsing them out of HTML texts by use of `<META>`-tags)

ENTRY IN NAME FIELD NOT UNIQUE WITH WEB PUBLISHING WIZARD

Description: When trying to insert documents with the Hyperwave Publishing Wizard I receive the error message "entry in name field not unique". I am sure that the document doesn't exist.

Solution: Are you using spaces in the names of the documents you want to insert? They cannot be handled by the Wizard because the name will later be used as the URL and cannot contain spaces for this reason.

MORE THAN ONE COMMAND IN HWMODIFY

Description: With the command line tool `hwmodify` it is possible to add and/or remove attributes to/from several objects in one run.

Solution: To address a whole collection and change all members of it, it is necessary to use the options `-coll` *collection_name* `-mult[iple] -recu[rsive]`. To apply more than one command you can combine them with a "\", for instance

```
hwmodify -coll mycoll -mult -recu -comm 'rem
PLACETemplate=mytemplate.html\add Rights=R:a'
```

5.3.3 MISCELLANEOUS

TCP/IP OR FILE SYSTEM CACHE PROBLEMS UNDER NT

If problems having to do with TCP/IP or file system cache arise (e.g. very long response times or physical memory runs out), you should consider installing the Microsoft hotfixes that are available for these operating system errors:

- `2gcrash`: Improves file cache policies and performance of very large databases (>200,000 objects).
- `iis4-fix`: This fix remedies the long delays or timeouts that can occur due to a race condition in Microsoft TCP/IP, especially on the first connection from the local machine.
- `pptp3-fix`: improves performance and stability of the TCP/IP stack especially over high latency networks, such as the Internet.

You can get these fixes online from Microsoft at:

```
ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-
postSP3
```

Please note that it is very important that you install the fixes in the order given above. Also note that neither Hyperwave Information Management nor Microsoft guarantees that these fixes are free of any unknown side-effects. Thus, if Service Pack 4 is available we recommend installing it instead of the fixes.

ADMINISTRATOR CANNOT LOG IN TO SERVER (UNIX)

Description: The server administrator cannot log in to Hyperwave Information Server.

Solution: Under UNIX, the system user name and password for the server are normally the same as those of the UNIX account under which Hyperwave was installed. However, if your system uses shadow passwords, this is not the case and you will not be able to log in to the server.

For the UNIX platforms the password depends on whether the UNIX system uses shadow passwords or not.

1. In most cases, the user name and password are taken from the UNIX account under which Hyperwave Information Server was installed.
2. In the case of UNIX systems with shadow passwords, the password field in `/etc/passwd` cannot be read, and thus the password of the UNIX account cannot be used. In this case, the user name is taken from the account and the password is "hwsystem". For security reasons it is recommended to change the password as soon as possible using **Site→Change Password**.

DEFAULT LANGUAGE GERMAN

Description: The default language for the Hyperwave Information Server is English. You may want another language to be the default language for users accessing your server.

Solution: `WAVEMASTER::DEFAULT_LANGUAGE = sp`

Note: The value of this variable must be one of the language abbreviations used in Hyperwave. See page 89 for a full list of abbreviations.

IDENTIFICATION AND OFFHOSTNAME

Description: Identification via web browsers always fails although you know that the user name and password are correct.

Solution: In order for the identification via web browsers to work correctly, the official hostname that appears in `.hgrc` must be a complete domain name or an IP address. In either case, it must contain **at least two dots**.

MICROSOFT'S ADD-ON TO IE3.02 DOES NOT WORK

Description: Is there a file upload mechanism similar to the one in Netscape also in the Internet Explorer?

Solution: Microsoft offers an add-on to IE3.02 that enables RFC1867-compatible file upload. However, it does not work (client waits indefinitely for a reply) because this add-on does not properly implement RFC1521 (as required in RFC1867) by violating the multi-part mime specification.

The close-delimiter of multi-part mimes is defined to:

```
close-delimiter := "--" boundary "--" CRLF
```

MSIE3.02 add-on omits the terminal CRLF, hence avoiding the MultiMime state machine of the gateway to proceed into the epilogue state which is to follow the close-delimiter. If the data is spooled (multimedia documents), this is not detected as the spooler waits for the missing characters. If the data is processed directly (text), the violation is detected and an "Invalid MultiMime" response is composed.

HTTP 1.1 PERSISTENT CONNECTIONS

Description: Are HTTP/1.1 persistent connections supported?

Solution: Yes, there is an option in the server configuration for this. You can set it via WaveSetup:

Choose the tab "Network" and check the field "Keep alive" in the area "Persistent Connections".

Manually:

Add the following line in `.db.contr.rc`

```
WAVEMASTER::KEEP_ALIVE=true
```

then kill the WaveMaster (it will be automatically restarted) or stop and start the server on NT.

IMAGES GET LOST AFTER DOCUMENT INSERTION

Description: In Hyperwave all objects are inserted into a database. For images to be referenced from within HTML texts they have to be inserted into the database, too.

Solution:

1. If you upload a document with an HTML editor, there should be an option to include the pictures in the upload process.
2. If you insert several documents consider uploading a whole subdirectory, including all images, by use of `hwupload`.
3. If you insert single documents remember to upload all referenced images and assign them the names under which they have been referenced.

IDENTIFICATION FAILS

Description: It is possible to use any web browser to access a Hyperwave Information Server. Depending on the browser you may, however, sometimes experience difficulties in the identification process. Under certain circumstances Netscape browsers will refuse your identification (although your password was typed in correctly) or will identify you as the same user you are already. Some instances of Internet Explorer remember your password as long as the program is running and you will not be able to change your identification or correct a wrongly typed password.

Error message: Identification fails. Access denied.

Solution: With Netscape you will be successful if you cancel the identification process once, or if you retry it several times. If you cannot re-identify with Internet Explorer you will have to exit the program and start it again to be successful.

GET HTML PAGE ON ACCESSING A COLLECTION

Description: In Hyperwave you can also get a normal HTML page (similar to an `index.html` on other WWW servers) instead of a collection listing, when accessing a collection.

Solution: Choose the text that you want to have displayed first, look at its attributes, choose "Edit attributes", select the option **PresentationHints** in the listbox and type in `FullCollectionHead` (exactly in this spelling and upper/lower case combination). If you want to edit something in this collection later, the FullCollectionHead is switched off when you switch to edit mode.

5.4 HELP

5.4.1 REFERENCES

If your problem is not in the scope of this manual, you may refer to other sources of information. Here are some references which may help you.

The documentation on Hyperwave consists of six parts:

- Hyperwave Installation Guide
- Hyperwave User's Guide
- Hyperwave Administrator's Guide
- Hyperwave Programmer's Guide
- Hyperwave HW API Definition
- Hyperwave Reference Guide

Any changes and updates of this documentation after going to press are available in the README files on the CD ROM with which this manual was delivered.

5.4.2 ONLINE DOCUMENTATION

All the addresses below refer to both of our WWW servers in Germany (`http://www.hyperwave.de`) and USA (`http://www.hyperwave.com`).

All five parts of the Hyperwave documentation are available online at `http://www.hyperwave.de/support/documentation`.

Apart from the manuals you will find there a list of Frequently Asked Questions, demo applications, up-to-date descriptions of the tools and more material which will help you if you need more information.

5.4.3 SUPPORT

PROBLEM DESCRIPTION FORM

If you cannot solve your problems with the help of the manuals or the online documentation, you may contact our support team. Please use the problem description form on the Hyperwave Information Server CD-ROM, or the online form at `http://www.hyperwave.com/support/request`. You can help us to locate your problem efficiently and give you the best assistance by providing as much information as possible about your environment, the steps that lead to the problem and the actions you have already taken. You should also always provide the appropriate log files.

We will especially need the following information:

1. architecture of the machine, operating system (on most UNIX systems you will get the appropriate information with `'uname -a'`)
2. content of the main log file (default: `$HOME/log/server.log`)
3. the log file and error file of the failing server (for organization of server directories and logfiles see page 6) If the appropriate logfile(s) are too large send approximately the last 100 lines (`tail -100 name_of_the_log_file`).

If you need to send log files, you may email them with your structured problem description directly to our support address `support@hyperwave.com`.

6 APPENDIX A

6.1 HYPERWAVE COPYRIGHT NOTES

The software on this CD is copyright protected. Some packages like Perl are protected by the GNU public license. This license requires the inclusion of the source code for the distributed packages. For these programs the source code can be found in the "<*path_to_CD*>`/unix/src`" directory, the source code includes also the exact text of the license.

In addition this CD contains the Acrobat Reader from Adobe to display the PDF documentation. The official copyright statement follows:

Acrobat © Reader Copyright © 1987-1996 Adobe Systems Incorporated.

All rights reserved. Adobe and Acrobat are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

Hyperwave is the trademark of Hyperwave Software Inc., the Hyperwave logo is a trademark of Hyperwave Information Management GmbH. Hyperwave Information Server, Copyright 1998 Hyperwave Information Management, GmbH. All rights reserved.

6.2 NETSCAPE COPYRIGHT STATEMENT

Portions © Netscape Communications Corporation 1996, All Rights Reserved

7 APPENDIX B

7.1 OFFICIAL HYPERWAVE COOKIE POLICY STATEMENT

The Hyperwave Information Server uses a session-oriented concept to manage state information. Because the underlying protocol (HTTP) is stateless, Hyperwave uses cookies to implement this session-oriented behavior. Although cookies are harmless and cannot be used to spy on users, there are often misgivings about how they work and what they do, causing users to disable their browsers for cookies. This practice leads to problems, as Hyperwave can then no longer map users to sessions. This in turn results in a loss of many of Hyperwave's features, including user identification, the ability to set language and quality preferences, and proper sequence handling.

Below is an explanation of how cookies work in Hyperwave.

What kind of cookies are used in Hyperwave and what do they do?

Hyperwave uses two kinds of cookies: session keys (sk) and Hyperwave session cookies. Both of these cookies are sent by the server to the client for each user session.

The session key cookie is a unique random number which is used as a key to a session pool, i.e. to determine the session pool the user belongs to so that he or she does not have to log in to the server at every access. There is neither a user name nor password in this cookie. The Hyperwave session cookie contains the actual session variables of the user, such as preferred MIME types, selected quality, current language, last collection accessed, and so on. Every time the user accesses the server, the client sends these cookies back to the server.

When are these cookies generated?

The sk cookie is generated on identification only. The Hyperwave session cookie is generated whenever there is a request which changes one or more of the values contained in it. For example, if you navigate to another collection, the LAST COLLECTION value in the Hyperwave session cookie is changed to reflect your state.

Where are the cookies generated?

The cookies are generated on the server and cannot get any information from your local hard disk.

Where are the cookies stored and when are they destroyed?

The sk cookie is not stored on disk, but rather the browser keeps the cookie in memory. It is valid only for the duration of the session.

The Hyperwave session cookie expires five years after it is created, and therefore the browser may save it on your hard disk. Note that this is only for reasons of convenience so that you don't need to adjust your settings every time you access the server.

What information is in the cookies?

The session key is a unique random number. Here is an example:

```
sk=00E7620419F38604
```

The Hyperwave session cookie can contain the last collection accessed, the last object accessed, the preferred language and the preferred MIME types (some of these fields are optional). Here is an example:

```
HyperwaveSession=LASTCOLLECTION=0xc0a80101_0x00000002,LASTOBJ=0xc0a80101_0x0000000a,L
ANGUAGE=en,PREFMIMETYPES=text%2Cimage%2Caudio%2Capplication/postscript%2Cx-
world%2Cmovie
```

Here are some references to general information about cookies:

```
http://fishcam.netscape.com/assist/security/faqs/cookies.html
http://www.cookiecentral.com/unofficial_cookie_faq.htm
http://www.netscape.com/newsref/std/cookie_spec.html
http://www.netscape.com/assist/security/faqs/cookies.html
```

7.2 THE COMMON LOGFILE FORMAT

The common logfile format is as follows:

```
remotehost rfc931 authuser [date] "request" status bytes
```

remotehost

Remote hostname (or IP number if DNS hostname is not available, or if DNS Lookup is Off.

rfc931

The remote logname of the user.

authuser

The username as which the user has authenticated himself.

[*date*]

Date and time of the request.

"*request*"

The request line exactly as it came from the client.

status

The HTTP status code returned to the client.

bytes

The content-length of the document transferred.

INDEX