How to get started with a Hyper-G Server
Version 0.93

Written by: Frank Kappe (fkappe@iicm.tu-graz.ac.at)
Walter Schinnerl (wschinn@iicm.tu-graz.ac.at)
October 20, 1995

Contents

1 Introduction

This document describes how to obtain, install, and configure the Hyper-G server software, and get started inserting information into the server. It is the first document a new Hyper-G server administrator should read to get started.

Hyper-G is a general-purpose, large-scale, distributed, multi-user, hypermedia information system. This document assumes that you have heard of and know about Hyper-G, and just want to sit down and install it.

1.1 Other sources of information

Throughout the documentation there are references to the Hyper-G ftp site at **iicm.tu-graz.ac.at**. Should you experience problems reaching iicm.tu-graz.ac.at, the **/pub/Hyper-G** directory is mirrored nightly to the following hosts:

ftp.tu-graz.ac.at:/pub/Hyper-G
elib.ZIB-Berlin.DE:/pub/InfoSystems/Hyper-G
ftp.ask.uni-karlsruhe.de:/pub/infosystems/Hyper-G
ftp.esrin.esa.it:/pub/Hyper-G
ftp.sunet.se:/pub/Networked.Information.Retrieval/Hyper-G
unix.hensa.ac.uk:/mirrors/Hyper-G/
ftp.cs.auckland.ac.nz:/pub/HMU/Hyper-G
ftp.utdallas.edu:/pub/Hyper-G
ftp.ncsa.uiuc.edu:/Hyper-G

A number of research papers are available that go into details of Hyper-G's design philosophy. In directory **Hyper-G/papers** you will find electronic versions of some of them. If you have never heard of Hyper-G before, we suggest you first look at files **report388.ps*** in that directory.

This document does not describe Hyper-G clients. Information on *Harmony*, the UNIX/X11 client for Hyper-G, can be found in **egmm94.ps*** in the directory mentioned above. Documentation on *Amadeus*, the PC/MS-Windows client, is available in directory **Hyper-G/Amadeus**.

Under **Hyper-G/doc/HTF.ps** you will find the description of HTF, the Hyper-G Text Format. This is a "must-read" before inserting text documents into a Hyper-G server. The protocol definition is contained in **Hyper-G/doc/Protocol.ps**, which is only of interest if you plan to write your own Hyper-G clients or tools that interface with the server directly.

1.2 New versions of this document

This is a not yet finished document. Some sections are missing. However, because of the rapidly increasing number of Hyper-G server administrators we have the feeling that we should get this out even in this state. We work hard to finish the documentation as soon as possible!

1.3 Feedback

If you have questions or comments about this document, please feel free to mail Frank Kappe, at fkappe@iicm.tu-graz.ac.at or Walter Schinnerl, at wschinn@iicm.tu-graz.ac.at . We welcome any suggestions, criticism, or postcards. If you find a mistake within this document, please let us know so we can correct it in the next version. Thanks.

2 Hardware Requirements

You must have one of the following machines and operating systems:

- SUN Sparc (under SunOS 4.1.3 or Solaris 2.2 or above)
- HP 700 series (under HPUX 9.01 or above)
- DECstation (under Ultrix 4.2 or above)
- SGI (under IRIX 3.0 or above)
- DEC Alpha (under OSF/1)
- IBM PC compatibles (under Linux)

The following platform is likely to be supported in the near future:

- IBM (under AIX)

To operate as a serious Hyper-G server, for HP's, SGI's or DEC Alpha's you should have at least 64 MB of RAM installed. For SUN's and machines under Linux we suggest at least 32 MB of RAM.

2.1 Space requirements

The Hyper-G server software itself consumes about 50 MB (depending on your platform). In addition, you will need enough space for the documents you are planning to serve. Double the space for text documents (required for the full text index), and add an extra 500 bytes overhead per document. For example, if you are going to serve 50,000 text documents of 1K each, plus 1000 images of 100K each, this sums up to 50,000 * 1K * 2 + 1,000 * 100K + 51,000 * 0.5K = 225.5 MB.

Allow more if you have lots of links between documents. In addition, you will need some free space for logfiles, temporary files, etc. Your mileage may vary!

3 Installing the Hyper-G Server

The installation procedure has been automated to the extent possible. Installation and software updates are performed with the help of the "Hyper-G Update Server". After the first installation (which requires some additional steps) it is sufficient to start a certain script and newer pieces of the Hyper-G software will automatically be copied over the Internet and installed. This script can optionally be started periodically (e.g., by cron) to ensure that you always run the latest software release. The modular design of the Hyper-G server even allows an upgrade of its code while it is running, without your users noticing it.

Together with the Hyper-G Server, a number of administration and utility programs, the Hyper-G Terminal Client, and additional documentation (man-pages) are installed and kept up to date.

The Hyper-G Server is distributed in binary form, and is available for the platforms mentioned under "Hardware Requirements". In order to install the software, you need to have root (super-user) privileges. However, once it is installed, the server could and should run as an ordinary UNIX user.

3.1 Getting hginstserver

First you may perform the following steps (you need root privileges to do this):

- Create a new UNIX user the Hyper-G server software will run as (say, *hgsystem*) and optionally a user group (say, *hyperg*). Put the home directory of this user on a file system with enough room (see 2.1), and give it a standard C-shell (*/bin/csh*) or an enhanced version of the C-shell (*/bin/tcsh*).
- Create a directory */usr/local/Hyper-G* with owner *hgsystem* and 0755 mode. This is the system-wide place for Hyper-G configuration files, documentation, etc. You don't need much room on the file system of this directory, as symbolic links to the actual places will be generated.
- Install *perl* (a popular script language) on the machine (your system administrator should have done this anyway). The *perl* interpreter must either be in the standard place (/usr/local/bin/perl), or its actual location must be in the path variable. If you don't have perl yet, it can be downloaded by anonymous ftp from prep.ai.mit.edu (and many other sites) in /pub/gnu/perl-4.036.tar.gz or /pub/gnu/perl5.000.tar.gz
- Get the perl script *hginstserver* by anonymous ftp from *Hyper-G/Server*, and put it into *hgsystem*'s home directory, make it owned by *hgsystem*, and 0755 mode.

3.2 Installing the software

To install the software, log in as user *hgsystem* (or whatever name you have chosen) and start the the *hginstserver* script.

The first time the script is run it will ask you a series of questions and create a file *.hgrc* in *~hgsystem* (user *hgsystem*'s home directory) and modify *.cshrc* so that *.hgrc* is included at the end of *.cshrc*. In *.hgrc* a number of environment variables are defined (you may edit this file after the installation has completed), including:

CPU: Your Platform (SUN4, SUN5, PMAX, HPUX9, SGI, etc.)

PATH: *~hgsystem/bin/scripts ~hgsystem/bin/$CPU)* is prepended to the existing path.

DIRServer: The directory where the low-level database server puts its files (default *~hgsystem/server*).

DIRfts: The directory where the full-text server puts its files (default *~hgsystem/ftserver*).

DIRdcs: The directory where the document cache server puts its files (default *~hgsystem/dcserver*).

DBHost: Name of the machine where the low-level database engine runs on (hostname of your machine).

HGHOST: Name of the machine where the Hyper-G server runs on (hostname of your machine).

GOPHHOST: Fully qualified domain name of your server machine, i.e., the name you want your machine to be known from the outside. This is used by gopher and WWW gateways when they are sending out gopher menus and URLs.

HGServerString: String describing this server (e.g., "University of Somewhere, SomeCity, SomeState"). Please use a reasonable name since this is how your Hyper-G server will be known to the world. This name is displayed by various clients!

The script will also create the directories *HTF, HTML, bin, bin/scripts, bin/$CPU, contrib, server, ftserver, dcserver, sgml, samples, doc,* and *man* under *~hgsystem*, with appropriate access permissions. It will also create symbolic links to these directories from /usr/local/Hyper-G. This is the standard place where other programs (e.g., the terminal client) will look for them.

After the previous steps (that are only necessary when installing for the first time) the binaries, scripts and configuration files for your platform are transferred over Internet, decompressed, and installed.

In order to keep the Hyper-G server software up to date, just start *hginstserver* again once in a while (or even let *cron* do that for you). This will contact the update server to check whether some pieces of the Hyper-G software have been replaced by newer versions, and install them if necessary.

NOTE:

hginstserver will also install a new version of itself (in *~hgsystem/bin/scripts*) if necessary. To avoid confusion, the old version that you got over ftp (in *~hgsystem*) should be deleted after the first installation!

Also if you have modified the *.hgrc* file save a copy in *.hgrc.sav* as the install scripts replace it each time it is run.

3.3 After installation

Let the system administrator compile *~hgsystem/samples/hgbindport.c* and put it to *~hgsystem/bin/$CPU/hgbindport* owned by root and with the setuid bit set (mode 04755).

This little program is used to obtain the privileged port numbers 418 (Hyper-G), 70 (Gopher) and 80 (WWW), and pass the openfile descriptors to another program that does not have root privileges. The program is delivered in source form so that the system administrator can check that it does not do nasty things as root. hgbindport opens a (system) port as file descriptor 3, changes its user id to hgsystem (the user who has invoked this command), and then exec's the appropriate program (e.g., gophgate). gophgate recognizes the open file descriptor and uses it automatically, instead of opening a port of its own.

The effect of this is that (after the installation of hgbindport) you don't need root privileges to start the gopher and WWW gateways, and the Hyper-G server. It therefore makes the system more secure (people don't have to trust the gateways etc.; they are started as ordinary user).

Once you have done this, you are all set to start your Hyper-G server. You may do so at any time by issueing the command *dbstart* (located in *~hgsystem/bin/scripts*) as user *hgsystem*.

If you want to stop the Hyper-G server you may perform this by issueing the command *dbstop*

However, in order to keep the Hyper-G server running all the time, you should add the following lines to your boot script (e.g., */etc/rc.local*):

```
#
# Start Hyper-G Server
#
if [ -f /usr/local/Hyper-G/bin/scripts/dbstart ]; then
    echo 'Starting Hyper-G Server'  >/dev/console
    su - hgsystem /usr/local/Hyper-G/bin/scripts/dbstart
fi
```

This starts the Hyper-G server control script (that fires off the individual server components) as user *hgsystem*.

You may want to change the default port numbers of the Gopher gateway (port 70) or the WWW gateway (port 80) to other port numbers (e.g. if you would like to run both the WWW server and the Hyper-G server on the same system). In such cases you should edit the file *~hgsystem/.db.contr.rc* and modify the numbers of <GOPHPORT> or <WWWPORT> appropriately (e.g., for running the WWW gateway on port 8000 change the line <WWWPORT> 80 to <WWWPORT> 8000) and then stop and restart the Hyper-G server.

If you don't want these gateways running at all just set the port number of that gateway to zero (e.g., <GOPHPORT> 0).

You may verify that the server is running by connecting to it using the terminal client (just start *hgtv*). You also may wish to install Harmony, the UNIX/X11 client for Hyper-G, if you have not done this already. Look under the **Hyper-G/Harmony** ftp directory to find out how to install Harmony on your machine.

4 Setting up Users and User Groups

When you start your server after you installed it, its store will be completely empty. You now need to decide how to fill it with information. This requires thinking about two things:

1. Who shall be able to insert information? Who shall be able to read it? This section covers how Hyper-G users and user groups are organized.
2. Where shall they insert it and how should your server's contents be organized and look to the user? The next section covers how to create a collection hierarchy.

The tool *hgadmin* has been installed by hginstserver (and so has the corresponding manual page, *~hgsystem/man/hgadmin.1*). This tool allows you to create Hyper-G users and user groups. To get started, it also allows you to create a pointer to the online help for the terminal client (which is fetched from the Graz University of Technology).

Before we show an example of how to set up a user, let us first describe the underlying concepts: Hyper-G supports a hierarchical model of users and user groups. Users identify themselves using username and password (this is optional; otherwise they are user "anonymous").

A user may belong to a number of user groups, and every user group may belong to a number of other groups. For example, user "fkappe" belongs to group "iicm" (his department), which in turn belongs to group "tu-graz" (the university).

On the other hand, every Hyper-G object "belongs" to a certain user, and may have access permissions attached (who may read/modify or delete this object; see section 6 "Defining Access Permissions"). For example, a certain document may be readable by all members of group "tu-graz".

In this case, user "fkappe" would be allowed to read it, because he is a member of group "iicm", which is a member of "tu-graz". In addition, Hyper-G users may have an associated "home collection", where they can create their own view of the information space, and store private objects.

Now let us do a little example. Suppose we want to have a Hyper-G account under the same name as the UNIX login account. Suppose this account name is "fkappe". In order to create a new user, invoke the *hgadmin* command, as user *hgsystem*. It should give a screen like this (for a full description of hgadmin see the manual page):

```
>>>>>>>>>>>>>>>> Hyper-G Administration-Tool <<<<<<<<<<<<<<<
        User: hgsystem(system)


1 New User
2 New Group
3 Edit User
4 Edit Group
5 Identify
6 Update Help

7 [q]uit

Enter your choice:
```

The line "User: hgsystem(system)" means that the Hyper-G server has identified the user *hgsystem*. This is a user that is created when the Hyper-G Server is first installed, and is the "super-user" for the

Hyper-G server. If "User: hgsystem(system)" does not appear you probably did not start *hgadmin* as user *hgsystem*, or you have changed *hgsystem*'s UNIX password in the meantime. In this case, use option 5 to identify yourself as *hgsystem* with the (old) password.

We choose option 1 to create a new user, and enter the username ("fkappe"). A new menu appears:

```
>>>>>>>>>>>>>>> New User <<<<<<<<<<<<<<<
        UserName        [fkappe]

1 Add Host
2 Add Group
3 Add Password
4 Add Password (encrypted form)
5 Add Description
6 Add Name of home collection
7 Rem Host
8 Rem Group
9 Rem Password
10 Rem Password (encrypted form)
11 Rem Description
12 Rem Name of home collection
13 Insert User-Object

14 [q]uit

Enter your choice:
```

We choose option 5 to add a clear-text description ("Frank Kappe") of the user, and become a member of the group "system" using option 2. Like user "hgsystem", group "system" is created when the server is first installed, and is the group of Hyper-G system administrators, i.e., every user that is a member of this group has system privileges (which means that no access rights are checked, and the user may use tools like *hgadmin*). Now we have to supply a password for this user. There are two ways to do this: When using option 3, enter the password in clear text. It is then encrypted and stored in the server in encrypted form. To log in later, one must supply the password again; it is then encrypted and verified against the encrypted password in the server.

For a UNIX user, there is a means to auto-identify using the UNIX username and password. In this case, the client automatically sends the UNIX username and encrypted UNIX password to the server. The server compares the UNIX and Hyper-G usernames and passwords and if they are identical, the user is automatically identified under the UNIX user name (no need to supply username and password again). This is essential for using non-interactive tools that manipulate information in the Server. For additional security, this is only possible from a list of hosts where this feature has been enabled. The auto-identify feature does not work if the client machine has a shadow password file.

To enable auto-identification, select option 4 from the menu, and supply the encrypted UNIX password. How do you get this password? Depending on the system, it is either stored in /etc/password, or in the passwd map of a NIS (YP) server. Either "grep fkappe /etc/passwd" or "ypcat passwd | grep fkappe" should give something like:

```
fkappe:Zdaxh8A9/EBxM:45:4:Frank Kappe:/usr2/users/fkappe:/bin/tcsh
```

The encrypted password is made up of the 13 printable characters after the first colon ("Zdaxh8A9/EBxM"). This is entered in *hgadmin* after choosing option 4 in the user menu. To enable auto-identification for the client host ("halley"), choose option 1 and enter "fkappe@halley". The hostname must be known to the Hyper-G server machine, otherwise you have to specify its internet address. Note that you can specify more than one password and host.

In addition, we choose to have a personal home collection (this is optional) for user "fkappe", and enter its name "~fkappe". We have to remember to create this collection afterwards.

When finished with the parameters of the new user, choose option 13 to really create the user object in the Hyper-G database.

Suppose we want to create new user group "iicm", and make "fkappe" a member of this group. Then we would choose option 2 from *hgadmin*'s main menu, enter "iicm" as the group name, add a description (option 2), and create it by selecting option 5 from the group menu. Then we would have to go back to edit the user "fkappe" and add the group "iicm".

We may test the auto-identification by contacting the Hyper-G server from the account (fkappe@halley), with the terminal client. In the top left corner of the client's screen, we should be able to read "fkappe(system)".

5 Creating a Collection Hierarchy

Next, you need to think about how to set up a collection hierarchy (you may think of a collection hiearchy as a directory hierarchy, or as a hierarchy of menus). Keep in mind that the collection hierarchy serves three main purposes:

- It lets the user navigate in a hierarchical fashion. Depending on the client, a hierarchy will be shown to the user in a graphical browser (Harmony) or as a sequence of menus (other clients).
- It lets the user specify the scope of searches. Searches can be limited to certain parts of the hierarchy (i.e., collections and their sub-collections, recursively).
- It lets you organize access permissions. You may specify that a certain user group may write information only into specific parts of the hierarchy, or that a part of the hierarchy shall only be readable by certain users or user groups.

In addition, users may be allocated personal "home collections", which is their default entry point and where they can organize their own view of the information space. In such a home collection it is often convenient to include the default home collection, i.e., the entry point for anonymous users, of this server.

Technically speaking, the collection hierarchy on a Hyper-G server is not a tree (where every item - with the exception of the root - has exactly one parent), but an acyclic directed graph (i.e., an item

```
     +--------------+
     |rootcollection|
     +--------------+
           /\
          /  \
         / +-------+
        /  |~fkappe|  . . .
       /   +-------+
      /      /  \
     /      /    \
   +----------+
   |~anonymous!  . . .
   +----------+
```

may be a member of a number of collections). Acyclic means that the collection hierarchy stored on a Hyper-G server cannot contain circles (otherwise certain algorithms that the server performs on the hierarchy would not work). The Hyper-G server is aware of this and won't let you create circles.

For referential purposes, every collection has to be given a unique name. This is not the title (which is displayed to the user, can be arbitrary long and need not be unique), but rather a short and unique name under which this collection can be referenced from the outside. You should try to choose readable names (some clients display the name of parent collections for query results), and keep in mind that the name is not (and can not be) changed afterwards, even when the collection is later moved within the hierarchy. The server won't let you create a collection with a name that already exists.

NOTE:

While collection names need to be unique they should not necessarly resemble the collection hierarchy used to access them, as this is very likely to change. They should instead have a name which is unique in a much wider, perhaps global, context.

When the server is first installed, the so-called *Root Collection* (with the name "rootcollection") is automatically created. Under this collection you should create a collection that will be the entry point of anonymous users (we suggest to use "~anonymous" as its name), and the personal home collections of the identified users (we suggest to use "~fkappe" for the home collection of user *fkappe*). This setup lets "~anonymous" also be a subcollection of "~fkappe" (see below), without generating a circle.

Under "~anonymous" you then create the "top-level menu" of your server, i.e., what should be visible for anonymous users. It is a good idea to place an item "about this server" (or similar) as the first item. By the way, the default order of items is by their sequence number (an attribute you may supply to any Hyper-G object), or else by the title. You may override this for any collection with the *SortOrder* attribute.

Individual collections are most easily created interactively using a Hyper-G client. For example, to create the "~anonymous" collection using the terminal viewer (hgtv) you would select the collection you want to insert the new collection to (at the beginning you don't have much of a choice; you have to select the rootcollection), enter the command "insert document", answer a few questions (title, owner, creation time, rights), choose "collection" as the type of the new object you want to create, enter "n" to the cluster question, and enter the name ("~anonymous"). Later on, you may modify or add attributes with the "modify attributes" and "add attributes" commands. The "move to collection" and "copy to collection" commands may be used to rearrange the hierarchy by moving (copying) subtrees around.

The above description is for the UNIX *htgv* client that is installed together with the server. There are similar procedures for the Amadeus (PC/MS-Windows) and Harmony (UNIX/X11) clients. Mass-insertion of collections can be performed using the *hginscoll (1)* command, which is also part of the server distribution. It is intended to be called by scripts (see the man-page for details).

6 Defining Access Permissions

Access rights are specified in the *Rights* attribute of a Hyper-G object (that includes documents and collections). 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.

By default (i.e., no Rights attribute present or a field is empty) only the author of the object has write and unlink permissions while all users have read permission. Write access implies read access. If the write field is present and the unlink field not, the users specified by the write field also have unlink permission, i.e., may delete the object. Specifying the unlink field allows collections that a (large) group of users may write documents into, but only a (small) group of users may remove documents from (e.g., this can be used for hypertext discussion).

In addition to the *Rights* attribute, access is controlled by the *TimeOpen* and *TimeExpire* attributes. After the date specified by *TimeExpire* the object becomes invisible for users other than the owner of the object and system users. (The object is not physically removed from the database, however.) Similarly, an object is invisible before the date specified by *TimeOpen* for users other than the owner of the object and system users.

Once again, read access is granted, if:

1. the user has write access (see below), or
2. the user has system privileges (i.e., is a member of user group "system"), or
3. the user is the author (owner) of the object, or
4. the time constraints are met (*TimeOpen, TimeExpire*), and
   (a) the read field of the object's Rights attribute is empty (i.e., unlimited read access granted), or
   (b) the user is a member of the users or user groups specified in the read field of the object's Rights attribute.

Write (unlink) access is granted, if:

1. the user has system privileges (i.e., 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 (unlink) field of the object's Rights attribute.

Write access for a document implies write access for all anchors attached (i.e., even for private anchors of other users).

6.1 Examples

Rights=R:g abcd Members of user group *abcd* have read permission, only the authors of the object has write permission (the write field is not present and thus the default value is used).

Rights=R:g a1 a2;W:g a1 User groups *a1* and *a2* have read permission, but only group a1 has write permission.

Rights=W:g b1 b2,u c1;U:g b1 All users have read permission (default; read field is empty), and groups *b1* and *b2* and user *c1* are granted write permission, but only the members of *b1* may also remove the object (or members of this collection if applied to a collection).

Rights=R:a Only the author has read (and write, by default) permission.

Rights=W:a This is the default setting that is used when the Rights attribute is not present: Everybody may read but only the author may write and unlink.

Attributes may be changed and added interactively using any Hyper-G client. If you have to change a number of objects in a similar fashion, you may use the *hgmodify (1)* command (see the man-page), either by itself or in a script.

7 Inserting Documents

Once you have decided about a collection hierarchy and inserted a few collections, you may want to insert information (i.e., documents) into the empty skeleton. You may either do this interactively using a Hyper-G client, or use the commands provided with your server installation to insert documents from the command line or from scripts.

7.1 Pointers to other Hyper-G Servers

Probably one of the the first things you should insert is a pointer (a reference) to the on-line help for the Hyper-G Terminal Viewer (hgtv). This documentation is stored on the Hyper-G server of the Graz University of Technology, and for reasons of consistency only a reference to it should be made in your local Hyper-G server. To make this a simple step, the *hgadmin* tool contains a function for that:

```
>>>>>>>>>>>>>>>> Hyper-G Administration-Tool <<<<<<<<<<<<<<<
        User: hgsystem(system)


1 New User
2 New Group
3 Edit User
4 Edit Group
5 Identify
6 Update Help

7 [q]uit

Enter your choice:
```

Just choose option 6 in the main menu. This will create (in your *rootcollection*) a collection *System Documentation* (with name *SysDoc*), under which two pointers to the remote online help are placed. From now on you should be able to use the on-line help from *hgtv* (press '?' for a start).

You may notice that above your *rootcollection* there is a (virtual) collection called "Hyper Root" (to reach it from the *htgv*, you have to use the "parents" command a few times). The children of this collection are the other Hyper-G servers your server knows about. In fact, your server should have information about all other servers (there is a special server-to-server protocol that ensures that), but it may take some time after you first started your server (up to a day or so).

You may navigate through the collection hierarchies of other Hyper-G servers from the *Hyper Root*, e.g., using *hgtv*. When you find a useful collection somewhere, you may use *hgtv's* "copy to collection" command to create a pointer to that information in your local collection hierarchy. You just need to enter the name of the collection where this copy should be put into. *Harmony* and *Amadeus* have the same functionality.

Currently, there is no non-interactive tool to insert pointers to other Hyper-G servers.

7.2 Pointers to Gopher/WWW/WAIS Servers

Pointers to Gopher, WWW, and WAIS servers, as well as telnet sessions can be created interactively using *hgtv*'s "insert document" command. After answering the standard questions, choose "remote" as document type, and then the protocol ("http" is for WWW objects). Depending on the protocol, you will have to supply additional information).

The *hginsdoc* tool allows to insert such pointer objects as well. For a concise description of *hginsdoc*, consult the manual (~hgsystem/man/hginsdoc.1), or type "hginsdoc -h" at the command prompt for a short description of its parameters.

7.3 Inserting text documents

Text documents can be inserted either interactively by using the Hyper-G clients *hgtv*, *Harmony* or *Amadeus* or by using the appropriate tools for inserting text documents from the command line or from scripts.

7.3.1 HTF text

Hyper-G Text documents have to be coded in Hyper-G Text Format (HTF). Its documentation has been placed under *~hgsystem/doc/HTF.ps*.

HTF is defined in terms of the ISO Standard Generalized Markup Language (SGML). HTF markup captures the **logical** structure of a text; formatting and layout is done later by the viewing application. Tags are used to specify headings, indentation, emphasis, itemised lists, paragraph breaks, and so on. Start tags are delimited by < and > and end tags are delimited with </ and >.
Ampersand (&) and less than (<) are special characters to SGML and must be escaped (entered as mnemonic entity references) as & (&) and < (<). Characters in the upper half of the ISO Latin-1 character set may be entered either as entity references: ä, ß, © (for example by 7-bit editors) or directly.

```
<TITLE>An Example Text Document in HTF
<H1>An Example Text Document in HTF<P>
The first tag of an HTF text is always the TITLE tag, which specifies the
document title in the database (it does not appear in the text itself).
The H1 tag introduces a level 1 heading in the text and is typically used
for the actual document title in the text.<L INDENT>
The L tag forces a line break and specifies the justification to be used
for subsequent text.<P>
Paragraphs are delimited by the P tag; a blank line also signifies a
paragraph break.

Text may be emphasised in three strengths from light to heavy:
<HP1>this is light emphasis</HP1>, <HP2>this is medium emphasis</HP2>,
and <HP3>this is heavy emphasis</HP3>.

<H2>Section 1<P>
Some of the more important features of Hyper-G are listed here:
<LI>
<IT>client/server architecture across the Internet;
<IT>multi-user;
<IT>both hierarchical and hyperlink structure;
<IT>integrated attribute and full text searches;
<IT>multilingual.
</LI>
```

Individual text documents can be simply created interactively using a Hyper-G client.
For example, to create the above text using the UNIX terminal viewer (*hgtv*), you would first identify yourself with username and password (command "identify") and select a collection where you want to insert the new text and where you have write permission. Then you enter the command "new text", answer a few questions (collection, new title, language) and choose an editor. The first and second line already contain the TITLE and H1 tag with your title, you should enter your HTF text beginning with the third line. After exiting the editor you confirm the question "Really save document? [y]" with "y" (yes) and your new text will be inserted.
By the way, if you want to use another editor as those given by the menu ("vi" or "emacs") you can edit the file *~/.hgedit.mnu* or */usr/local/Hyper-G/hgedit.mnu* and add your preferred text editor.
With *Amadeus*, the PC/MS-Windows client, similar commands are available. For identification you use the command "Identify" in the "Administration" menu. The "New" command in the "File" menu allows you to insert different documents (text, image, ...) into the database.

The *hginstext* tool allows to insert already existing HTF text files.
First of all you have to ensure that the auto-identification from your actual account and host is enabled. Enter "hgadmin" and check the name behind "User:". If there is your username the auto-identification works, if there stands "User: anonymous", choose option 5 (Identify) to identify yourself, then option 3 (Edit User) with your username, check if your actual host is already added and compare the encrypted Hyper-G and UNIX passwords (see section 4).

Before you really insert the text file into the Hyper-G database, the command *hgparse* can be used to verify that the HTF document is correct and to check the visual appearance of the text. If for example the name of the HTF text file is "example.htf" you enter "hgparse example.htf". If the file doesn't contain valid HTF, self-explanatory error messages are produced. (See also manual page *~hgsystem/man/hgparse.1*)

When your HTF text is correct, you use the *hginstext* command to insert the text. For this you have to know the name of the collection ("name of the parent"), where you want to place the text. For example, to insert the above text into the collection with name "~anonymous" you would enter "hginstext -pname '~anonymous' example.htf". (see the man-page for details).

Mass-insertion of text documents can be performed by using the *hginstext* command called by scripts (see section 8).

If your text document is rather complex, you may also consider inserting it in Postcript format (see "Inserting Non-Text Documents").

7.3.2 Other text formats

One of the other text formats, which also can be used in conjunction with Hyper-G, is **RTF** (Rich Text Format). RTF can be exported by most modern word processors.

Individual RTF files can be simply inserted interactively using the Hyper-G client *Amadeus*. You click on "New" in the "File" menu and select the "Text" command. Then a dialog box will be opened where you enter the collection name and title, choose the document in the file chooser and set the language. In our case you would choose RTF as object type.

The *rtf2htf* tool allows to convert Rich Text Format (RTF) to Hyper-G Text Format (HTF), (see the manual page for details). Having converted the text to HTF format, you continue with "Inserting HTF Text" (see 7.3.1).

If you want to insert plain **ASCII** text, HTF's PLAIN tag can be used. If the title is followed by the <PLAIN> tag (instead of <H1>) the rest of the text is rendered in a fixed font as it is typed in, with respect to line feeds. If the ASCII text contains < or & characters you should use the <VERBATIM> tag or escape these characters.

```
<TITLE>Example Plain ASCII Text<PLAIN>
This is usually used for tabular data, pieces of source, etc.
It is not appropriate for standard flowing text.
<VERBATIM>
Between the VERBATIM tags < or & characters have not to be escaped.
</VERBATIM>
```

7.3.3 Text links

In Hyper-G a link consists of the *source anchor*, the visible representation of the link and the *destination anchor*, which can be used to specify only a part of a document as destination of the link. Anchors may appear anywhere (including plain ASCII text), and they may be arbitrarily nested and overlapping. To achieve this, there are actually two tags: <AS> (Anchor Start) and <AE> (Anchor End). The required ID attribute specifies which <AS> and <AE> tags belong together and form one anchor.

Source anchors have an additional DHINT (destination hint) attribute. A source anchor definition looks like this:

```
<AS ID="0x00000000" DHINT="collname lg:title">anchor text<AE ID="0x00000000">
<AS ID="0x00000001" DHINT="collname keyword">anchor text<AE ID="0x00000001">
<AS ID="0x12345678" DHINT="collname">anchor text<AE ID="0x12345678">
```

where *collname* is the name of the (parent) collection the destination document should be searched in, *lg* is the language (e.g., 'en' for English) and *title* is the title of the destination document. If the title within the destination collection is not unique, you may also specify a unique *keyword* that has been attached to the document. If the destination is a collection, you can just specify the *collname* of this collection. The values of the ID attributes are not relevant as long as they are unique within the text. Don't forget to surround the attributes by quotes!

Should the target document not exist (yet), an 'open' link is inserted into the database. Open links are not shown when looking at the source document. Everytime a new document is inserted, it is matched against the open links for the corresponding collection. If a matching link is found, the link is closed (i.e., it is now visible and selectable from the source document). This feature allows to insert easily a number of documents with circular references in any order. Also, when a document is deleted the links pointing to it are marked as open, i.e., they cannot be selected any more by the source documents. However, when a document with the same title is inserted into the same collection, the links are restored again.

Destination anchors have an additional KEY attribute in the <AS> tag. A destination anchor may specify an area of a document. The default destination anchors for text, image , movie and audio are the entire document, for PostScript the first page and for a scene the default camera position. The following is a valid definition of a destination anchor:

```
<AS ID="0x00000001" KEY="en:TheWholeList">
<LI>
<IT>Item 1
<IT>Item 2
<IT>Item 3
</LI>
<AE ID="0x00000001">
```

The KEY value is exported as the title of the destination anchor, so that you can search for it, put it into collections or clusters and attach links to it. The values of the ID attributes are not relevant as long as they are unique within the text.

Links in text documents can also be created interactively using a Hyper-G client.
To create links using *hgtv* you edit the text and mark the areas where you want to have a link to emanate from (the source anchors, usually one or more words) by bracketing them into << and >>. After exiting the editor, you will be asked for every source anchor to navigate to the corresponding destination object and execute the "make link" command by pressing `^L` (Control-L).

Harmony can also be used for inserting links interactively. First you mark the visible area of the link, then you select "Define As Source" in the "Anchors" menu. Now you have to specify the *destination anchor* by selecting the object (e.g., in the Session Manager), then go to the "Anchors" menu, select "Define As Destination" and create the link.

With *Amadeus* you can use the "Make Link" command in the "Edit" menu to insert a link into a text document.

7.3.4 Inline images

The <INLINE> tag is used to define an inline image in the document. The image is inserted at the tag position and aligned with the base line of the current text line. The following graphic formats are supported: GIF, TIFF, JPEG and XBM.

To display an image inside a text document, it has to reside somewhere in the Hyper-G database as an image document. The mechanism to reference the image is the same as used for the anchor tags. The <INLINE> has two attributes, the ID and the DHINT attribute, which have the same meaning as in the anchor start tag <AS>. An inline image definition looks like this:

```
<INLINE ID="0x00000000" DHINT="collname lg:title">
```

where *collname* specifies the collection name containing the image, *lg* is the 2 letter language specifier and *title* is the title of the document which is the image. An image with the title specified must be found in the said collection otherwise an 'open' link is inserted into the database.
The ID value is expanded out automatically once the image has been matched via the DHINT argument.

Example of Use:

First insert the image (see section 7.5) into the database into any collection (for example, into an auxiliary collection 'Inline Images') as an own document. If you don't want that collection to be visible to general public, set the access rights of this collection to 'R:a' (readable only by author). But be sure that the inline image itself has the same access rights as the text document! Then make a link to the image in every place within the text in which you want it to occur.

To create inline images using *hgtv* you insert a new text with <INLINE> tags or edit an existing text by adding these tags.

```
This is a picture of the grand canyon:
<INLINE ID="0x0" DHINT="Inline en:grand_canyon.gif">
```

When inserting a new inline image the ID attribute can be set to any value (e.g., "0x0", "0x12345678").

Inline images can be created easily using *Harmony*. In the *Text Viewer* you select first the place where the image shall appear using the left mouse button and then 'Anchors/Define as Inline Source'. After this you select the Image in the *Session Manager* and then 'Anchors/Define as Destination'. You commit with 'Create Link' in the *Link Creator*.

7.4 Creating Clusters

While a (default) collection is displayed as some sort of menu and lets the user choose one of its members, a cluster is displayed by visualizing all of its children simultaneously.

A cluster is described as a "Multimedia Document". It pools together many single media items and presents them as a single complex document. This makes it possible to show a text while showing an image and while playing a sound, for example.

However, the cluster concept is also used by clients to choose one of a number of documents which only differ in their language. This allows you to define explicitly a sub-group of documents to be visualised for certain language conditions. In this case a cluster is used as a "Multilingual Document".

The exact rule for visualizing a cluster is:

1. All language-independent members of the cluster are visualized. Language-independent objects have more than one title attribute (in different languages).

2. From the remaining language-dependent members, one of each document type (i.e., one text document, one picture and so on) is chosen (based on the user's language preferences) and visualized. Language-dependant objects have exactly one title attribute.

On some (rare) occasions it may be necessary to visualize a number of language-dependent documents of the same document type in a cluster (e.g., two images in two languages each). To achieve this effect, the author can group related documents of different language in language- independent sub-clusters.

Though you also can put collections inside clusters, doing this provides no special capability within Hyper-G at the moment. The collections are not accessible unless you use the "children" command to see the list of objects within a cluster.

Individual clusters are most easily created interactively using a Hyper-G client.
In the Hyper-G terminal viewer for example, to create a cluster use the command "insert document" . You follow the same steps as for creating a collection except that the prompt asking if you are creating a cluster should be answered "yes". Once created you can insert documents as if the new cluster is a Collection.

With *Amadeus* you use the "New" command in the "File" menu and select "cluster" to insert a new cluster.

Mass-insertion of clusters can be performed using the *hginscoll* command with option -c (for cluster).

7.4.1 Multilingual documents

A multilingual document consists of a cluster and more than one language-dependent document of the same document type.
A text, which you want to present in different languages (e.g., in English and German), is a simple example for a multilingual document. In this case, you would first create a cluster with an English and a German title and then insert into this cluster the English text with an English title and the German text with a German title.

To give a document (e.g., cluster) more than one title is very easy. By using the "add attributes" command in the terminal viewer (*hgtv*) simply add a title with a different language prefix. Titles start with a two letter prefix which specifies the language component of the title. This is followed by a colon and the title string. For example

```
Title=en:Language English
Title=ge:Sprache Deutsch
```

You can also use *Harmony* or *Amadeus* to add titles in different languages by using the command "Attributes" in the "Edit" menu.

Any document that has two or more lines of title is regarded as a language-independent document (any document must not have two titles with the same language prefix).

7.4.2 Multimedia documents

Another function of clusters is to manage multimedia documents.
An example for this might be presenting information about a person. Suppose that this cluster contains a picture, some text about the person, and perhaps a recorded message. If this cluster were accessed all the single media items would be displayed or played together.
In such a case, you would first create a cluster with one title and after this insert into this cluster the image, text and sound document (see section 7.5).

If we take the example above and extend it to include two or more pictures of the person, then we first insert the additional images into this cluster. After this we have to make all the images within this cluster language-independent by giving them a second title with a different language prefix. This guarantees that more than one document of the same document type (in our case "image") is visualized.

7.5 Inserting non-text documents

The following non-text documents can be inserted into Hyper-G:

- Image (GIF, JPEG and TIFF)
- Movie (MPEG-1)
- Sound
- 3D Scene
- PostScript

These documents can be created interactively using *hgtv* with the command "insert document". After answering the standard questions, choose one of the document types "Image", "Movie", "Sound", "Scene" or "PostScript", and then enter the local filename of the document.

The *hginsdoc* tool allows to insert such non-text documents as well. For a concise description of *hginsdoc* consult the manual page.

With *Amadeus* you use the "New" command in the "File" menu to insert different document types into the database.

8 Scripts that insert information

While it it possible in principle to insert and maintain a body of hypermedia information using only a Hyper-G Client (*hgtv*, *Harmony* or *Amadeus*) information providers who contribute large amounts of information routinely should write scripts or programs that manipulate information in the Hyper-G server. This section gives an overview of the "foundation" commands necessary to write such scripts. Note that in order to use these commands the auto-identify feature has to be enabled for your account!

In Hyper-G, every object has to be member of at least one *collection*. The *hginscoll (1)* command is used to create such collections within other collections. It may also create clusters (i.e., collections that display their contents simultaneously rather than offering a menu).

The *hginstext (1)* command may be used to insert (a number of) text documents in HTF format. Use the *hginsdoc (1)* command to insert other document types (Image, Movie, Sound, Generic) in clusters.

The *hginfo (1)* command may be used to extract attributes (e.g., title, keywords, author, access rights) of individual objects, while the *hgmodify (1)* command can be used to change them.

Two commands are available for (mass) deletion of objects: The *hgdelobj (1)* deletes individual objects, children of a given collection, or any object which matches a given query. The *hgerase (1)* command may be used to remove recursively a collection hierarchy.

8.1 Example Perl script

Supposing you have the following telephone list file and you want to insert the telephone database into Hyper-G with one text document for each person:

```
# LAST NAME (30)              FIRST NAME (20)     TELEPHONE NR. (20) #
#=============================|===================|===================#
Holden                        Anna                2245-765
Vanhorn                       Steven              9987-321
Manners                       Dan                 8722-562
Connor                        Barbara             4726-988
```

The following perl script is an example of how to use *hginscoll* and *hginstext* for inserting the telephone database file with filename "`telephone.xmp`" into a Hyper-G server.

```
#!/usr/local/bin/perl

#
#     Example perl script for inserting a telephone database file
#     into a Hyper-G server, using the common Hyper-G UNIX tools.
#

$hghost = "hyperg";      # Hyper-G server name
$author = "hgsystem";    # Hyper-G author
$parent = "~anonymous"; # Name of parentcollection
$lang   = "en";          # Language

$phonename  = "Phone.Numbers"; # Name of the phone numbers collection
$phonetitle = "Phone Numbers Collection"; # Title of the collection
$phonefile  = "telephone.xmp"; # Name of inputfile
```

```
# Hyper-G system call to insert the phone numbers collection
system ("hginscoll -r $hghost -n $parent -N $phonename -A $author -T '$phonetitle' -L $lang");

open (phonefile);

while (<phonefile>) {
    if (/^#/) {    # Comments, go to next line
        next;
    }
    if (/^\s*$/) { # Forget blank lines
        next;
    }
    $line = $_;    # Current input line

    # Assign the first 30 characters to $lastname
    $lastname = substr ($line,0,30);
    # Strip off all whitespaces from the end of the string
    $lastname =~ s/\s*$//;

    $firstname = substr ($line,30,20);
    $firstname =~ s/\s*$//;

    $telephone = substr ($line,50,20);
    $telephone =~ s/\s*$//;

    print "Inserting \"$lastname, $firstname\" in Hyper-G\n";

    # String for insertion in Hyper-G is built
    $layout = "<TITLE>$lastname, $firstname\n";
    $layout.= "<H1>$lastname, $firstname<P>\n";
    $layout.= "<DL>\n";
    $layout.= "<DT>Lastname<DD>$lastname\n";
    $layout.= "<DT>Firstname<DD>$firstname\n";
    $layout.= "<DT>Phone Number<DD>$telephone\n";
    $layout.= "</DL>\n";

    # Open a pipe to tool "hginstext" to insert text of format HTF
    # to the Hyper-G server
    open (HG, "| hginstext -hghost $hghost -pname $phonename -lang $lang -author $author");
    # Print the prepared string to the pipe
    print HG $layout;
    # Close the pipe
    close (HG);
}

close (phonefile);
```

9 Register your Server

This section has deliberately been put at the end, because you can do the world a favour in NOT announcing your server until you have some useful information in there.

10 Miscellaneous

Please mail us at fkappe@iicm.tu-graz.ac.at or wschinn@iicm.tu-graz.ac.at if any part of this document is confusing or incorrect. We depend on feedback from readers in order to maintain this document!

Best of luck with your new Hyper-G Server!

Cheers, Frank and Walter