Computational Chemistry Network Services and User Interfacing. 2

H. de Hilster, A. H. M. Thiers, and J. H. Noordik*

CAOS/CAMM Center, University of Nijmegen, Toernooiveld, NL-6525 ED Nijmegen, The Netherlands

Received January 21, 1998

This paper describes the recent developments in the CAOS/CAMM Center General Menu System (GMS). GMS has been under development since 1988 and was reported before in 1993. The spectacular growth of WWW based services and the proliferation of personal workstations and computer networks urged us to completely overhaul the system to comply with these new possibilities. X11 graphics options have been added, and a WWW version, in which network services can be accessed from a Web page, has been developed. Emphasis has been, as before, on the ease of maintenance of large database and program collections for the network services provider and use for the end-user scientist.

INTRODUCTION

In a previous paper[1] the development and the usage of the CAOS/CAMM Center General Menu System (GMS) have been described. It was shown that centralized maintenance of large program and data libraries for Computational Chemistry is very cost-effective for both the data provider and the end-user, if structured through a single user interface with data access over the network. The rapid growth of (bio)chemical data resources, which prohibits distribution over desktops, the development of client/server computing, and the rapid increase in WWW based services over the past few years confirm the above observation. Centralized data management with *host access* and/or terminal emulation via *Internet technology* is going to account for the majority of (secure) data access operations in the (near) future. Browsers are or will become available for almost every platform including network computers, and users will prefer the familiar WebPages instead of standard terminal screens. Approaches such as GMS and completely new concepts such as the CACTVS[2] system have emerged to address the needs for cost-effective solutions for the retrieval of chemical information. The first GMS release was a *SMG* based VMS application.[1] The second release, for a UNIX system as main entry point to the services, was *curses* based. Both resulted from the design requirement that the majority of our users accessed the Center's services with only VT100/Tek4010 capable terminals. That this situation would change rather rapidly was to be expected. We saw graphical user interfaces, based on X11, being added to many of our chemical applications programs. We also see many of our end users slowly upgrade to graphics capable hardware and software and Web browsers. These developments justified the definition of a new GMS release that still would support character-based terminals, while simultaneously incorporating the current and future new options.

METHODS

System Design. The list of demands for an updated GMS clearly calls for a *client/server* solution with optional character based terminal support. With these boundary conditions the following main design features were formulated:

−separation of the basic menu system, in which programs and data are organized, and the actual user interface

−development of a graphical user interface to the menu system, with local look and feel and access to local files

−support for character based terminals

−easy installation by the end user

−all menu system versions (ASCII, X11, other) should use the same Menu Definition Files.[3]

At first sight, this specification pointed to a dedicated client/server version of GMS in which the end user would see a familiar screen and could specify *local files* as input to *host* applications. However, such a solution would have serious (maintenance) consequences.

−Central (host system) support for clients on a wide variety of platforms and for several flavors of UNIX, Windows3.1, Windows95, and Macintosh would be required.

−The end user would have to download and install (client) software, to access host services. For many sites this requires the aid of a system administrator. This is an option for *intranet* services on single applications (e.g., MDL's ISIS Draw[4] and Beilstein's Crossfire client[5]) but much less suited for access to continuously changing suites of programs and data.

−Modifications in the menu system would require the user to reinstall the menu client, undoubtedly leading to reluctancy in the development and extension of GMS.

Careful consideration of these consequences made the development of a dedicated client/server version of GMS less attractive. As a compromise between the ASCII menu and a dedicated client, for users running graphical applications on the host system and thus having X11 capabilities anyway, an X11 version of GMS was developed. But no access to local files could be provided. Moreover, the list with requirements with for a new GMS had to be extended with and

−the possibility for automatic upgrade of the client software

S0095-2338(98)00326-6 CCC: $15.00 © 1998 American Chemical Society
Published on Web 09/01/1998

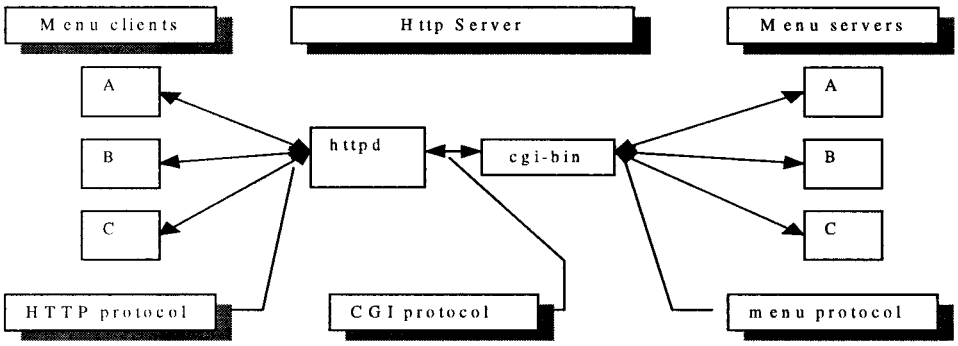


Figure 1. Schematic of the CGI approach to the GMS menu server.

Table 1. Example of Resources, Filters, and Their Relation to Requested URLs

url	filter	function
/private, /menu	BasicAuthFilter	Request a password from the user
/	LogFilter	Logs access
/help	CountFilter	Counts the number of requests

url	resource	function
/	FileResource	Returns a single file or directory listing.
/cgi-bin	CgiResource	Starts external cgi-bin processes
/menu	MenuResource	Returns menu pages, and starts menu actions

url	handled by resource	filters called
/gms/index.html	FileResource	LogFilter
/help/tutorial.html	FileResource	LogFilter, CountFilter
/menu	MenuResource	LogFilter, BasicAuthFilter

Table 2. Minimal Server Resource That Returns the Current Date/Time

```
import java.util.Date;
import caos.net.http.*;

public class DateResource extends BaseResource {

  public DateResource() {}

  public HttpReply get(HttpRequest request) {
    HttpReply reply = new HttpReply(request, HttpReply.OK);
    HtmlPage html = new HtmlPage();

    html.append("The current date and time is:");
    html.append(new Date());
    reply.setContent(html);

    return reply;
  }
}
```

−a very general client that could also be used for other client interfaces.

Several recent developments made the realization of a client/server GMS, fulfilling all of these requirements, possible. The enormous success of the World Wide Web has brought about a *universal client*, the Web browser. In its early phase the Web browser was not versatile enough to be used as a client interface to the GMS, but enhancements such as forms and tables turned the Web browser into a really general GUI.

Using a Web browser as client:

−The client code is sent to the user when needed, eliminating the need to install a separate client.

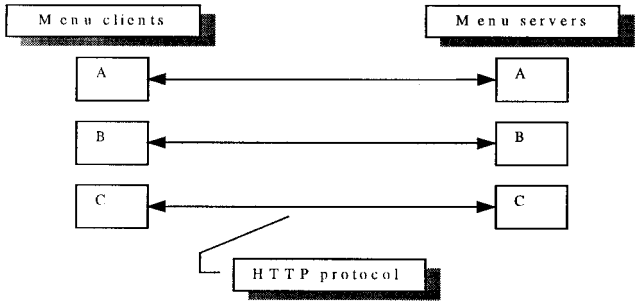
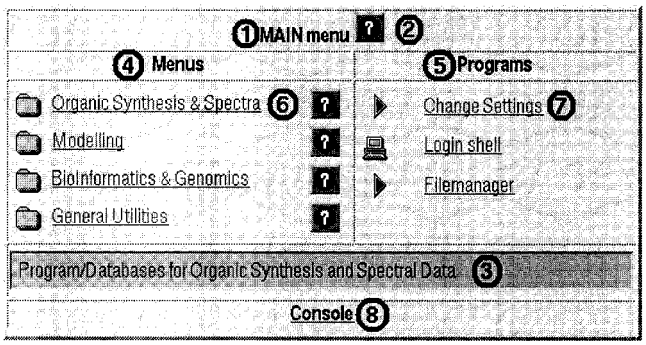


Figure 2. Schematic of an integrated HTTP- and GMS menu server.



1	*title*	
2	*helpfile*	*linking to the html help page*
3	*helpline*	*helpline is shown with the mouse is over the link*
4	*submenus*	
5	*actions*	
6	-	*name of link is the title of the submenu*
7	-	*icons indicate the type of link; url, form or direct action*
8	-	*link to full console text*

Figure 3. Layout of the menu as it appears in a Web page.

−Upgrading the client code is no longer a problem.

−One application will run on all the different client machines.

The following paragraphs describe the architecture and the coding of the GMS Web menu.

The CAOS/CAMM Center Web Menu System. The basic components for a client/server application are the server, the client, and the protocol between the client and the server. Having selected the Web browser as client, the protocol had to be the HTTP protocol. For the server side this left us with different options.

−A normal HTTP server, extended with menu server functionality. The extension being either through (fast) CGI[6] or by means of (dynamically linked) external code.[7]

−A dedicated menu server, speaking the HTTP protocol.

A menu system like GMS, tarts programs and accesses files for a specific user; therefore, it must run with the same credentials as the end user. An http daemon normally runs as a specific user process and serves all clients (Web browsers), consequently the first server approach requires an extra process and an extra protocol. See Figure 1 for a schematic of the involved processes and protocols.

From a design point of view, the second approach is much simpler because the menu server speaks the http protocol. Figure 2 illustrates this design.

This approach is equivalent to the use of http server extensions for the menu services, invoking an http server for each client process. This imposes the following requirements on the http server:

•a modular lightweight http, optimized for size instead of fast http transport

•easily extendable with different functionalities

•startable numerous times

•runable as any user

Although there are several modular httpd servers available (Apache,[8] Jeeves,[9] Jigsaw[10]) none of them are intended or easily changed to support the above requirements. In particular the notion of one server per user, different http addresses (different port numbers) for each server invocation, and changing the credentials of the server after http authentication have been addressed in the development of our new publicly available http server called WWWaiter.[11]

WWWaiter a Modular http Server. An http server handles requests from an http client (Web browser) and returns an http response. This response might be an html page, a binary file, a directory listing, etc. With extensions the server can be made to behave such that, e.g., a request triggers a database query, returning a query result response formatted in an html table. Different requests are to be handled by different server resources, e.g., a *file resource*, a *directory resource*, or a *database resource* depending on the requested URL.

Besides resources that handle specific task in the http server, filters can be placed anywhere on the server tree. The http server calls filters before and after the resource handles the request. A filter can return a response in which case the resource is never called. A good example of a filter is an access filter. If the user has not supplied a password, the filter returns an authentication response. If the user has supplied a password the filter has no effect. Table 1 summarizes some resources and filters and their function. Table 2 shows, as an example, the coding of the DateResource. All http server code is written in Java.

The GMSMenuResource. The GMSMenuResource implements the menu system. Menus are specified in simple, easy modifiable ASCII menu definition files (MDF). MDFs describe the menu layout and the programs/program parameters. The first GMS implementation used a fixed file format, but soon some dynamic behavior was needed. So in the second implementation the scripting language Tcl was used as the menu definition language (MDL).

The following text shows, as an example, the MDF for the main menu. Its appearance in a Web page is illustrated in Figure 3.

```
MENU root {
  -title        "MAIN menu"
  -helpline     "The CAOS/CAMM Main Menu"
  -helpfile     {gms/gms.html}
  -submenus     {caos camm cammsa mainutil}
  -actions      {settings loginshell fileman}
}

ACTION settings {
  -title        "Change Settings"
  -helpline     "Change your settings"
  -url          "/menu/settings"
}

ACTION loginshell {
  -title        "Login shell"
  -command      "/bin/tcsh"
}

ACTION fileman {
  -title        "Filemanager"
  -url          {/menu/fm/$CWD}
}
```

The following text shows the MDF for an action "gelview" from the GCG[12] package. Its appearance in a Web page is illustrated in Figure 4. The input parameter decriptions are translated to html FORM elements. After the user submits the form, the input is checked at the server (e.g., the input file must be specified), and the variables in the command string are replaced with the actual values of the input

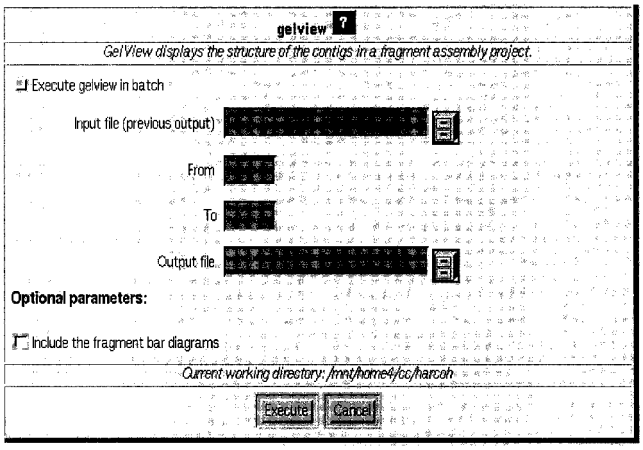


Figure 4. Layout of the action as it appears in a webpage.

parameters.

```
ACTION gelview {
  -title "gelview"
  -command {"gelview -INFILE1=%s -BEGIN1=%s -END1=%s -%s -out %s" {input
from to inc output}}
  -helpfile "gelview.html"
  -helpline "GelView displays the structure of the contigs in a fragment
assembly project."
  -controls {
    CHECK batch {0 0} "Execute gelview in batch" 0 {" " "batch"}
    EDIT input {30 1 20} "Input file (previous output)" "" {required}
    EDIT from {30 2 4} "From" ""
    EDIT to {30 3 4} "To" ""
    EDIT output {30 4 20} "Output file" ""
    STATIC msg {0 5} "Optional parameters"
    CHECK inc {0 6} "Include the fragment bar diagrams" 1 {"DUMMY" "BAR"}
  }
}
```

The MDF files for the large GCG suite are automatically generated from the GCG configuration files (see next paragraph for implementation details).

The File Manager. The icon after the input fields of type FILE is a link to a Web based file selector. This selector can be used to select files remotely (at the site of the Web server) but also allows for file upload from the user's machine to the remote machine. The file manager (see Figure 5) removes the strict separation between local and remote file space. The same filemanager, when not used as a file selector, can be used to view files and manage files and directories.

Depending on the configuration of the Web browser, the output of programs can be viewed directly in the browser. E.g., the Chime plugin[13] can be used to view xyz, pdb, and mol2 files. Downloading of files can be achieved by the "Save" option of the browser.

IMPLEMENTATION DETAILS

Tcl as the MDF Language. Our earlier menu systems used a fixed file format for describing the menus and actions. This often was experienced as too constraining. The current system uses Tcl in several ways to combine simple declaration and flexibility.

The MDF files are Tcl scripts defining menus and actions, but they may contain any Tcl code, allowing dynamical creation of menus and actions. Second, instead of simple strings one can also use dynamic tcl scripts, e.g.

```
-title {#! return "hello $env(USER)"}
```

This is especially useful for the action command. Instead of a fixed command with parameter substitution, a script can check the parameters and act on them. The command parameters are global tcl variables.

```
-command {#! if {$check == "y"} {
                gms_process "bfg_check"
              } else {gms_process "bfg_nocheck"}
         }
```

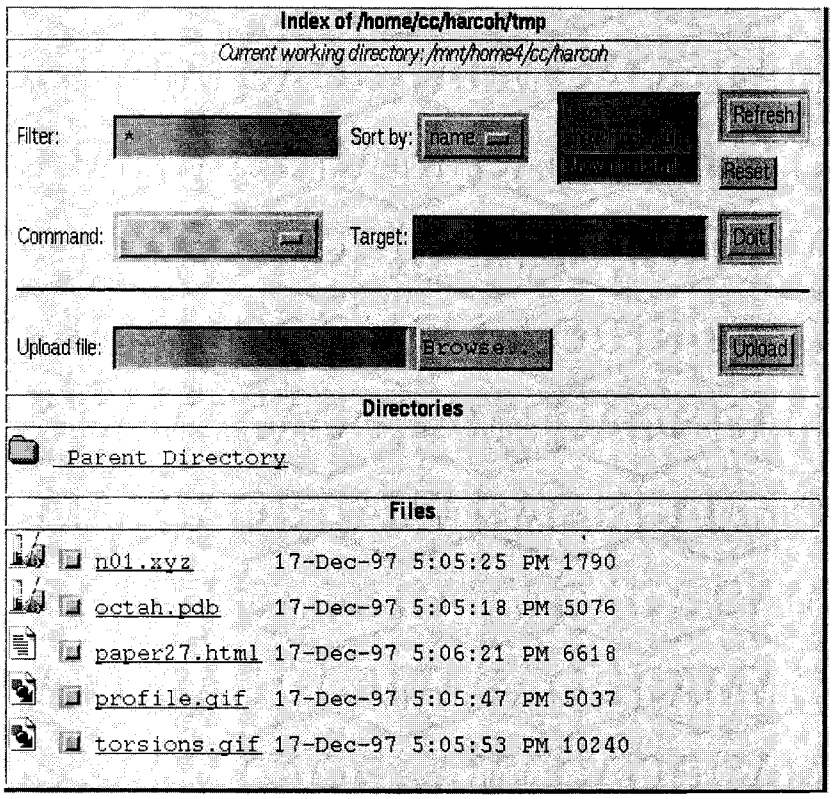


Figure 5. The file manager.

An example of the usage of dynamic menu creation is the GCG MDF file. The GCG[12] package comes with configuration files for the wpi X interface, describing each command and its parameters. The GCG MDF file uses these configuration files to generate equivalent MENU and ACTION commands when the GCG MDF file is loaded.

```
ACTION simplify {
  -title {simplify}
  -command {#! gcg_process simplify}
  -controls {#! gcg_form simplify}
  -helpfile {simplify.html}
  -helpline {Simplify lets you reduce the number of symbols in a
sequence.}
```

Menu Server Startup. The startup of the private http server is rather complex, because each process must use a different port number and should run initially as a root process so it can change to the users credentials.

•A user starts the menusystem by clicking on the link http://www.caos.kun.nl:8888/menu, which is a redirecting resource.

•This resource starts a private menu server through the normal cgi-bin mechanism and expects an URL as result on stdout, e.g., http://www.caos.kun.nl:2498/menu/root.

•Then this redirect resource sends an HTTP redirect answer with the new URL and also sends the URL as a cookie.[14] This cookie is used later to redirect the client to the previously started menu server instead of starting a new server on each request. This way one can use absolute URLs (http://www.caos.kun.nl:8888/menu/caos/lhasa) and still refer to each individual menu server.

The private http server is started as a root process and has a basic authentication filter followed by a "change uid" filter. The authentication filter blocks every request until the user has provided a valid username/password, and the change uid filter then changes the server process from a root process to a user process.

CONCLUSION

The Web implementation of GMS has been operational at the CAOS/CAMM Center for almost a year now. It has proven to be an excellent access mechanism to a large variety of computational chemistry tools. The familiarity of the user interface and the seamless integration of the HTML documentation provided by the Center has made the access much more user-friendly.

Although the development of the dedicated http server WWWaiter[11] required considerable coding effort, the server framework can easily be used for other Web interfaces. Especially the fact that the http server can run with the user's credentials, enabled new applications such as the file manager resource that can view, copy, delete, and upload files to the user's file space.

Further development of these server resources will extend the capabilities of our Web server and menu system. Currently we are, e.g., developing a sequence selector for the Center's bioinformatics services. Like the file manager manipulates files, this resource is intended to select an amino acid or nucleotide sequence for further processing.

ACKNOWLEDGMENT

GMS has been developed with financial support of EMBnet, the European Molecular Biology Network.

REFERENCES AND NOTES

(1) Thiers, A. H. M.; Leunissen, J. A. M.; Miller, T. M.; Schaftenaar, G.; Noordik, J. H. Computational Chemistry Network Services and User Interfacing. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 8−862.

(2) Ihlenfeldt, W. D.; Takahashi, Y.; Abe, H. Dataflow Processing in a Global Networked Context: A Solution for the Computational Methods Pool Management Problem. *Proceedings of the 28th Annual Hawaiian International Conference System Sciences*; Hunter, L.; Shriver, B. D., Eds.; IEEE Computer Society Press: Los Alamitos, U.S.A., 1995, Vol. V; pp 227−235.

(3) Menu Definition or MDF files have been defined in ref 1. These are simple ASCII files, specifying the appearance of the menu and the actions on button activations or mouse clicks.

(4) Isis Draw. MDL Information Systems Inc. San Leandro, CA, U.S.A.

(5) Crossfire. Beilstein Informationsysteme GmbH, Varrentrapp Strasse 40−42 D60486, Frankfurt, Germany.

(6) CGI: Common Gateway Interface. The http server spawns an external program to extend its normal functionality. Fast CGI: The http server connects to an already running external CGI program. This saves startup time.

(7) Some http servers allow external code to be embedded in the http server itself. Either by means of a scripting language or by (dynamically) linking external object code.

(8) Apache. One of the most frequently used http servers. See http://www.apache.org/.

(9) Jeeves. An http server developed by Sun Microsystems. See http://www.javasoft.com/products/java-server/webserver/.

(10) Jigsaw. An http server developed by W3C. See http://www.w3c.org/Jigsaw/.

(11) Additional information about GMS is available at http://www.caos.kun.nl/gms. WWWaiter and GMS source code, example MDF files and installation instructions, are available from the authors on request (email: harcoh@caos.kun.nl).

(12) Wisconsin Package Version 9.0, Genetics Computer Group (GCG), Madison, WI. See http://www.gcg.com/.

(13) Chime plugin by MDL Information Systems Inc. See http://www.mdli.com/.

(14) Cookies can be used by http servers to store information at the client (browser) side. See RFC 2109.

CI980326F