HOST LINKS

GWEB™

Web Browser Front-End for DKU VIP7700-7760 VIP7800 IBM3270 IBM5250 Emulations

http://www.gar.no/hostlinks/

GALLAGHER ROBERTSON

Version 6.5

GALLAGHER & ROBERTSON AS, Kongens gate 23, N- 0153 Oslo, Norway
Tel: +47 23357800 • Fax: +47 23357801
www: `http://www.gar.no/`
e-mail: support@gar.no

Contents

Installation

The G&R emulations and gateways are independent programs, but part of the *G&R Host Links* product set available on all major UNIX/Linux platforms. Many of the products are also available for Windows servers. For details on platforms supported, software delivery and installation refer to the *Host Links Installation and Configuration* manual.

Host Links Product Overview

Terminal environment

Host links products that run on UNIX or Linux servers with a terminal driven user interface include emulators and concentrators, as well as various utilities.

- G3270 provides synchronous IBM3270 functionality. G3270 emulates IBM LU type 2, including base and extended colour together with extended highlighting.

- Qsim provides synchronous Questar terminal functionality. Qsim simulates all Questar models, including the DKU7007, DKU7107, DKU7105 and DKU7211 (Mono, four colour A/B and seven colour modes are supported). It also simulates the VIP7760 and the VIP7700.

- V78sim provides Bull VIP78xx (BDS) functionality. V78sim emulates all models of the VIP7800 family; the actual reference is the BDS7. All visual attributes including colour are supported.

- Pthru provides transparent VIP7800 visibility to Bull mainframes for users with asynchronous VIP7800 terminals or emulators. The terminals are used in text or forms mode.

Server environment

Host Links products that run on UNIX, Linux or Windows servers.

- Ggate is a transparent gateway to the Bull native network. It avoids all need for Front-ends (MainWay/Datanet) or other gateways. It can be used to connect G&R/Glink (for Windows or Java) emulators or any of the emulators, concentrators, network printer emulators and file transfer clients/servers in the Host Links product set. It also supports third party clients using the TNVIP, TN3270, TN3270E and standard asynchronous Telnet protocols.

- Gweb provides a web browser interface to any host application that is otherwise accessible using the *Host Links Qsim*, *V78sim*, or *G3270* emulations.

- Gspool is designed to run as an unattended process and accept transparent print output from any type of host application (GCOS8, GCOS7, GCOS6, IBM) that normally sends print data to network printers (ROPs), or to a remote spooling system (DPF8-DS). On the Gspool system the print may be directed to a physical printer or to the local spooling system. Gspool operates in different modes, Connect mode, Terminal Writer mode, DPF8 mode, SNM mode, IBM mode, TN3270 mode and TN3270E mode.

- GUFT is a G&R implementation of the Bull UFT file transfer protocols. It enables transfer of data files between Host Links and GCOS systems over a DSA network.

- Gproxy is a network management program used for supervision, management, load balancing and license sharing of G&R *Host Links* applications. Gproxy can be set up as a freestanding monitor program and/or report generator in a small network, or play a bigger role in a larger network.

- Gsftp is a transparent gateway between two different File Transfer protocols: FTP (RFC 959) and SFTP (the SSH File Transfer Protocol). The purpose is to present a seamless integration between the two protocols, with automatic conversion.

Scope of the product

Functionality

Gweb provides a web browser interface to any host application that is otherwise accessible using the *Host Links Qsim*, *V78sim*, *G3270* or *G5250* emulations. Please refer to the product manuals for details of these emulations.

Please do not confuse Gweb functionality with the fully functional terminal session obtained when using *Glink*. Gweb allows a standard browser (Netscape, Microsoft Internet Explorer, Firefox, Opera, etc.) to initiate a session from a web server to the mainframe using the appropriate emulation. The session between the browser and the web server is not persistent, but Gweb still offers almost the same performance as a fully functional and persistent terminal session direct to the application.

Gweb makes mainframe applications developed using standard terminal interfaces accessible to large populations who do not have an appropriate emulator. Gweb is the ideal way of making mainframe information available to the general public without specialized web development. Your existing applications run immediately, and you can make new applications for access from the Web simply by developing a few display transactions using your standard development tools and mainframe programmers to generate screen images for IBM3270, IBM5250, VIP7700, VIP7760, VIP7800 or DKU terminals. Your existing and new applications are immediately available to any web browser provided you install Gweb on the web server, and add the necessary configuration parameters for the GCOS host and application to the configuration file. Gweb enhances the screen images by automatic application of basic facelifting, and you can enhance them further by applying wallpaper and HTML headers and footers that show your company logo or other images.

Gweb can connect using any G&R line handler to reach Bull or IBM mainframes using Bull DSA, DIWS, RFC1006, X.25, Telnet, TN3270, TN5250 and TNVIP links.

Gweb operates on most UNIX and Linux platforms as well as Windows servers. It can work together with any World Wide Web server that supports the Common Gateway Interface (CGI) for external applications. This includes most popular UNIX/Linux World Wide Web servers such as Apache, Netscape or NCSA, and Microsoft's Internet Information Server on Windows servers. Note however that best performance is achieved by using the Gweb standalone HTTP server, GwebS, included with the product. It is also by far the simplest to set-up, and occupies less space on your disk than any other HTTP server.

Gweb includes development tools that allow you to further enhance the screen images from the host applications by including your own HTML for presentation of individual screens. Gweb builds a library of the host screens after basic facelifting has been applied, and this library of HTML pages can be edited using any HTML development tool on a screen-by-screen basis. The development package includes a macro language for accessing areas or fields in the host screen image, and returning values to be used in the screen fields when the reply is transmitted to the host. The macro language also allows conditional display of HTML sections depending on the content of areas or fields on the host screen.

Network connections

Gweb communicates with the Bull mainframes using native DSA either directly or via a TCP/IP connection to a *G&R/Ggate* gateway. The DSA connection can be over an OSI transport connection to a MainWay or Datanet; or direct, avoiding the front-end, to the ISL Ethernet adapter on GCOS7 or the FCP7/FCP8 FDDI adapters on GCOS7 or GCOS8.

Gweb can also connect over TCP/IP using native DSA with RFC1006 as the transport layer into a MainWay with an ONP (where RFC1006 support is standard) or direct into the FCP7/FCP8 adapters if RFC1006 support is installed (NetShare). Direct RFC1006 connections can also be made into the GNSP of new GCOS8 systems, or the Windows personality of new GCOS7 Diane systems.

Gweb/7800 and Gweb/DKU support the TNVIP protocol and can make TCP/IP connections via the TNVIP servers of both GCOS7 and GCOS8.

Gweb/3270 and Gweb/5250 support TN3270 and TN5250 for TCP/IP connection to IBM hosts.

Run-time licenses

In order to run Gweb, the following license keys must be present in your `/usr/gar/config/licenses` file:

`basic`	For the base G&R run-time system
`gweb`	For Gweb Professional Edition, *or*
`gwbdiane`	For Gweb Diane Edition, *or*
`gwbgnsp`	For Gweb GNSP Edition, *or*
`gwbpthru`	For Gweb HTML Pass-through for Web7/8
`ssl`	For the SSL-enabled version of GwebS *(optional)*

The license is for a number of simultaneous sessions. Your license should be sized for maximum load expected.

Installing Gweb on your World Wide Web server

UNIX/Linux web server

Step 1: Supported platforms

Supported 32 bit UNIX/Linux platforms

386l2	Intel Pentium PCs, Linux kernel 2.x
386so	Intel Pentium PCs, Solaris release 8
hpp11	Hewlett-Packard, HP-UX 11.x
ppca5	Bull Escala, AIX 5L for Power V5.1/5.2
rs6a4	Bull Escala/Estrella, IBM RS/6000, AIX 4.3
spaso	Sun SPARC, Solaris 7/8

Supported 64 bit UNIX/Linux platforms

i64l2	Intel Itanium 2, Linux kernel 2.x
x64l2	AMD/Intel x64, Linux kernel 2.x
i64h1	Hewlett-Packard, HP-UX 11i v2

UNIX 32 bit deliveries by special request

The following platforms can be delivered by special request, at a cost based on time and materials and time frame depending on available test and qualification resources.

`386v3`	Intel Pentium PCs, SCO UNIX V5
`alpdu`	Alpha, Digital Unix 4.x
`hppu1`	Hewlett-Packard, HP-UX 10.x
`rs6ai`	Bull DPX/20 and IBM RS/6000, AIX 4.2

Step 2: Delivery

These instructions assume you have all the distribution files placed in a directory on the UNIX/Linux machine (e.g. `/tmp/gweb`). You should have received the following files, which are used only during the installation:

Name	Description
`Install.640`	Installation script.
`pppcccoo.640`	one or more ZIP archives containing the software.
`Unzcccoo`	Unzip program.
`srb.640`	Software Release Bulletin.
`licenses`	Optional, license keys to give access to the software.

Note: The *cccoo* matches the code in the supported platform table. *ppp* is a software component code; *gwb* for Gweb.

Gweb requires a license key to run. If your distributor did not include a licenses file in the delivery then you will have to manually enter licenses after the installation process. Instructions on how to do this should be delivered with the license keys.

Step 3: Unpacking and installing the software

For a new installation we strongly recommend that you create a new UNIX/Linux user id and group id for G&R software. An obvious choice of names would be:

```
username=gar
groupname=gar
```

The default system directory is `/usr/gar`. The installation script will give you a choice of selecting another directory as the default system directory, but you should note that if you do that, you must define a `GAR_SYSDIR` environment variable that points to this directory.

To make your installation as simple as possible we suggest that you log in or `su` to the root user, create the directory `/usr/gar` and make the `gar` user and `gar` group the owner of it with the `chown` and `chgrp` commands before you run the installation script.

After logging in as, or `su`ing to, the `gar` user: `cd` to the directory that contains the delivery files. Start the installation script with:

```
sh install.640
```

After installing Gweb, you should find the following new files in your Host Links binary directory:

```
gweb
gwebs
gwebss
gwebedit
3270web
5250web
7800web
dkuweb
```

You should also find a new directory in your Host Links system directory called `html`, and a directory under that called `gweb`.

Step 4: Installing the Gweb CGI script

Your web server has a separate directory for Common Gateway Interface (CGI) scripts. This directory is located inside your server's main directory (often `/local/www`), and is usually called `cgi-bin`.

To install Gweb for use with a UNIX/Linux web server, you will have to link or copy the `gweb` CGI program into your web server's CGI script directory.

In the UNIX/Linux shell, change the current directory to the server's CGI script directory. Then enter the command:

```
ln -s /usr/gar/bin/gweb .
```

If you cannot or will not configure your web server to allow links out of the CGI script directory you have to copy instead:

```
cp /usr/gar/bin/gweb .
```

Step 5: Adding a link to the Gweb HTML files

The next thing you do is to add a link in your web directory structure to the directory where Gweb stores its HTML files. These are the files that users download into their browsers to make new connections using Gweb.

Sample files which demonstrate how to set up a Gweb front page are installed together with Gweb in the `/usr/gar/html/gweb/` directory. There are different ways of making this directory visible to your web server, depending on which server you use. The easiest way is to make a UNIX/Linux file system link from somewhere in the web server's main document structure to the Gweb HTML directory.

For example, if you decide to store your network's HTML files in the directory `/local/share/www/docs/`, and you would like your users to access Gweb with the URL:

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

you should establish a link from `/local/share/www/docs/` to the directory `/usr/gar/html/gweb/` by entering the following command in the UNIX/Linux shell:

```
ln -s /usr/gar/html/gweb /local/share/www/docs/gweb
```

Windows web server

Step 1: Supported platforms

Supported 32 bit platforms

386pc	Windows 2000, Windows XP, Windows 2003

Supported 64 bit platforms

i64pc	Intel Itanium 2 Windows XP, Windows 2003

Step 2: Delivery

A Windows delivery is a single self-extracting exe file.

640_en32_xxxxxxxx.exe	xxxxxxxx is <complete> or <hostlinks>

Step 3: Unpacking and installing the software

Host Links products for Windows are delivered as self-extracting archives. When run they will automatically extract to a temporary directory and start the installation program.

You must also obtain a license file from your distributor. If you already have Host Links running on the system, please close all of these applications before you start the new installation. You can use the Gservice administration program in the G&R program group to terminate Gservice and all the G&R background server programs. You must terminate any client programs manually. Please make sure that you are logged in as a Windows administrator.

Installation steps:

1. Double-click the installer executable.
2. You will be prompted for a destination directory and a default will be suggested. The directory will be created if it doesn't exist.
3. You will be asked to choose a 'Full' or 'Custom' installation. Please choose a full installation unless you have received specific instructions for making a custom installation.

If you are not updating an existing installation you will be asked for the license file. Contact your distributor if you don't have a license file.

When the installation program has finished installing the files, choose Setup to create the G&R program group with icons.

After installing Gweb, you should find the following new files in your Host Links binary directory:

```
gweb.exe
gwebs.exe
gwebss.exe
gwebedit.exe
3270web.exe
5250web.exe
7800web.exe
dkuweb.exe
```

You should also find a new directory in your Host Links system directory called `html`, and a directory under that called `gweb`.

The following instructions apply for any web server. Some specific instructions for installing and setting up Gweb with the Microsoft Internet Information Server are included.

Step 4: Installing the Gweb CGI script

Your web server has a separate directory for Common Gateway Interface (CGI) scripts. This directory is located inside your server's main directory, and is usually called `cgi-bin`.

To install Gweb for use with a Windows web server, you copy the Gweb executable file, `gweb.exe`, from the Host Links binaries directory into your web server's CGI script directory.

Find your web server's CGI script directory. Then use the File Manager, Windows Explorer, or command prompt to copy the file `\gar\bin32\gweb.exe` to your web server's CGI directory.

Microsoft IIS users: if you are using Microsoft's Internet Information Server, CGI scripts are usually stored in the `\inetpub\scripts` directory.

Microsoft IIS v6 users: if you are using Microsoft's Internet Information Server version 6, you need to add `\gar\bin32\gweb.exe` to the list of allowed "Web Service Extensions".

Step 5: Adding a link to the Gweb HTML files

The next thing you do is to add a link in your web directory structure to the directory where Gweb stores its HTML files. These are the files that users download into their browsers to make new connections using Gweb.

Sample files that demonstrate how to set up a Gweb front page are installed together with Gweb in the `\gar\html\gweb\` directory. There are different ways of making this directory visible to your web server, depending on which server you use. Please read your web server's documentation for help doing this.

Microsoft IIS users: When using Microsoft's Internet Information Server, this can be done in the "Internet Service Manager" program. Launch the "Internet Service Manager" from the IIS group in the Start menu. Select the web service from the list of services. From the "Properties" menu, select "Service properties..." Select the "Directories" tab in the dialog box that appears, and click the "Add..." button. In the "Directory" field at the top of the dialog box, enter the path to Gweb's HTML files (usually `c:\gar\html\gweb`). Then select the "Virtual directory" radio button and enter the alias you want to use for Gweb's files in the "Alias" field. For instance, to be able to refer to Gweb's HTML files with the URL `<http://www.mycompany.com/gweb/>`, you would enter `/gweb` in the Alias field. Check off the "Read" checkbox at the bottom of the window, and click OK. Gweb's HTML files are now available to the outside world!

Installing GwebS

The GwebS standalone HTTP server eliminates the need for a third party web server. Gweb works identically in both modes (Stand-alone with GwebS and as a CGI program with a third party server). However, as GwebS is a separate server, it needs configuration on how it should operate, the port on which to listen, and so on.

GwebS configuration

The configuration file for gwebs is called `gwebs.XXX` and resides in the `/usr/gar/config/default` directory on UNIX/Linux platforms, and `C:\gar\config\default` on Windows servers. The `XXX` part, called the "mode ID", is by default `DEF`, but different configuration files can be selected using a parameter to gwebs, where `gwebs -mi ibm` instructs GwebS to use a configuration file called `gwebs.ibm`.

An installation of Gweb places an example `gwebs.def` file in the default directory. Modify this to suit your needs.

Example gwebs.def configuration file

```
*** gwebs.def configuration file for
*** web host "gweb.gar.no"
Host http://gweb.gar.no
Port 80
Logging extended
Alias /cgi-bin/gweb     gweb
Alias /cgi-bin/nph-push cgi   nph-push
Alias /ph14.gartest     web8  browser
Alias /glinkj           http://www.gar.no/glinkj
Alias /                 ../../html/
Index index.htm
```

The directives are explained in detail below.

Gweb server configuration file directives

Any line not starting with an alphabetic character [A-Z, a-z] is considered a comment line.

Port port-number

This defines the TCP/IP port number on which GwebS listens for connections. The default is the standard HTTP port, 80. Note that some systems require you to run with supervisor privileges if you want to use a port number below 1024.

Listen IP-addresses or *host names*

This defines the TCP/IP addresses on which GwebS listens for connections. The default setting is that GwebS listens on all interfaces on the machine. You may want to use this directive in multi-homed environments in order to control which web server corresponds to different IP addresses. Either dotted quad-type addresses or DNS resolvable host names separated by spaces can be used.

DocumentRoot path

This defines the root directory for serving documents. The default value is a directory name html immediately subordinate to the top-level Gweb or Host Links installation directory; e.g. `/usr/gar/html/`

Host hostname

This is the host name to use for self-referencing URLs. In some circumstances, the server needs to send redirect instructions back to your browser, and this redirect must contain a complete URL. Default value:

```
Host http://ip-address[:port]
```

Vhost hostname configfile

This defines the name of a host-based virtual server configuration. A modern browser will include the DNS name of the URL in the request. If this name matches one of the hostnames configured with the Vhost directive, the request will be processed according to the settings in the specified configuration file instead of the main configuration file.

A configuration file for virtual hosts uses the exact same syntax as any other configuration file, except that some directives are ignored. The ignored directives are Port, Listen and all SSL settings. Example:

```
Vhost subserver.gar.no subserver.gwebs.def
```

In this example, the configuration file `subserver.gwebs.def` contains settings for requests to the web `site http://subserver.gar.no/...`

Content-type MIME-type Extension

This directive defines the WWW content type for files with various file extensions. You may override the defaults, and extend the table with your own settings. The default values are specified in the table below.

```
Content-Type    text/xml                    xml
Content-Type    text/plain                  txt
Content-Type    text/plain                  text
Content-Type    text/html                   htm
Content-Type    text/html                   html
Content-Type    text/css                    css
Content-Type    image/png                   png
Content-Type    image/gif                   gif
Content-Type    image/jpeg                  jpg
Content-Type    image/jpeg                  jpeg
Content-Type    image/jpeg                  jpe
Content-Type    image/png                   png
Content-Type    video/mpeg                  mpg
Content-Type    video/mpeg                  mpeg
Content-Type    video/mpeg                  mpe
Content-Type    video/avi                   avi
Content-Type    video/x-ms-wmv              wmv
Content-Type    text/vnd.wap.wml            wml
Content-Type    image/vnd.wap.wbmp          wbmp
Content-Type    application/x-glink         glink
```

```
Content-Type    application/x-javascript       js
Content-Type    application/x-java-jnlp-file   jnlp
Content-Type    application/x-x509-user-cert   usr
Content-Type    application/x-x509-ca-cert     crt
Content-Type    application/x-x509-ca-cert     ca
Content-Type    application/x-pem-file         pem
Content-Type    text/x-server-parsed-html      ssi
Content-Type    text/x-server-parsed-html      shtml
```

Any file with an extension not defined in a Content-Type statement will be tagged with the MIME type `application/octet-stream`.

There is a special content type you can define to designate CGI programs:

```
Content-Type    script/cgi                     cgi
Content-Type    script/cgi                     exe
```

Logging Off|Standard|Extended

This directive controls the generation of a web server-like log file. The format, Common Log Format (CLF), is exactly the same as used on all major web servers, which means that it can be used with standard web server log analysis tools.

The CLF file is called `access.XXX` (where `XXX` is the server's mode id) and contains one separate line for each request.

By default, this file is limited to a size of 1MB. If it grows larger, it will be renamed to `access.XXX.0` and a new `access.XXX` will be started. Any existing `access.XXX.0` will renamed to `access.XXX.1` and so on. The highest number of log file generations is 9.

A line is composed of several tokens separated by spaces:

```
host ident authuser date request status bytes
```

The Extended CLF (also called the NCSA combined format) has two more fields at the end:

```
referrer user-agent
```

If a token does not have a value then it is represented by a hyphen (-). The meanings and values of these tokens are as follows:

Name	Description
`host`	The fully qualified domain name of the client, or its IP number if the name is not available
`ident`	If IdentityCheck is enabled and the client machine runs identd, then this is the identity information reported by the client. For GwebS, this field is always empty
`authuser`	If the request was for an password protected document, then this is the userid used in the request. For GwebS, this field is always empty
`date`	The date and time of the request, in the following format: `date = [day/month/year:hour:minute:second zone]` `day = 2*digit` `month = 3*letter` `year = 4*digit` `hour = 2*digit` `minute = 2*digit` `second = 2*digit` `zone = +' |-') 4*digit`
`request`	The request line from the client, enclosed in double quotes (")
`status`	The three digit status code returned to the client
`bytes`	The number of bytes in the object returned to the client, not including any headers
`referrer`	(extended format only) If you had followed a hyperlink from some HTML page to make this request, this field contains the URL of the page where the link was -- i.e. where you came from
`user-agent`	(extended format only) The User-Agent identification of the browser that made the request

The maximum size of the access log before it is rotated is 1MB. You can append the keyword `:size=n[B|K|M]` to the `Logging` directive to control this. The `n` is a decimal size of the maximum allowed access log in Megabytes (same as `nM`). The qualifiers `M`, `K` and `B` denotes Megabytes, Kilobytes and Bytes, respectively. A size of zero disables autorotating.

Examples:
`Logging extended :size=512K` denotes autorotating at 512 Kilobytes.
`Logging extended :size=0` disables autorotating

Index Index-file-name

This directive contains the name of a default file that the web server should give you if you requested a directory. You can specify multiple entries here; for example:

```
Index index.htm
Index index.html
```

If you request a directory (such as http://gweb.gar.no/), then GwebS will first look for:

```
http://gweb.gar.no/index.htm
http://gweb.gar.no/index.html
```

If neither file exists GwebS will check to see if directory listing is enabled (see ***Enabling directory listing***). If directory listing is not enabled GwebS returns HTTP status 403 (Forbidden) to the browser.

Alias Virtual-part Physical-Path

This section describes the mapping between a request for a virtual part and a physical path. The Physical-part represents the replacement part of the Virtual part.

Mapping a request to a physical location

```
Alias / U:\www\docs\                (Windows)
Alias / \\machine\share\docs\       (Windows)
Alias / /share/www/docs/            (UNIX/Linux)
```

Here, requests to anything starting with a slash are mapped to a local path according to the rules above. Since these are effectively catch-alls (all URL to your server will begin with a slash), you would normally define a directive such as these as the last one. In the examples above, a request for:

```
http://your.host.name/gweb/config.htm
```

causes the files in the table below to be sent to the browser:

```
U:\www\docs\gweb\config.html             (Windows)
\\machine\share\docs\gweb\config.html    (Windows)
/share/www/docs/gweb\config.html         (UNIX/Linux)
```

If the file does not exist, an error message (HTTP error code 404) will of course be returned.

```
Alias /news/   http://www.timesonline.co.uk/  (external redirect)
```

This causes any request that starts with `/news/` to result in a redirection to an external URL:

```
http://www.timesonline.co.uk/                (external redirect)
```

Enabling directory listing

If the URL delivered to GwebS references a directory, Gweb will look for an index page, with the name(s) specified in the Index directive. If the index page is not found GwebS will by default return HTTP status 403 (Forbidden) to the browser. If you want Gweb to return a directory listing, you can use a mapping directive of the form:

```
Alias / U:\www\docs\          :listing=1   (Windows)
Alias / \\machine\share\docs\ :listing=1   (Windows)
Alias / /share/www/docs/      :listing=1   (UNIX/Linux)
```

Mapping a request to the Gweb subsystem

```
Alias /cgi-bin/gweb gweb
```

The Virtual-part represents the apparent location of Gweb. You would use e.g. `/cgi-bin/gweb` here, since this is the default location for CGI programs, but GwebS does not use the CGI technique, so you can use any valid value. The Physical-path must be the key word **gweb**. Whenever GwebS receives a request that begins with `/cgi-bin/gweb`, control is transferred to Gweb.

Mapping a request to the CGI program handler

```
Alias /cgi-bin/nph-push cgi   nph-push
```

The Virtual-part represents the apparent location of a CGI program.

The Physical-path is the key word **cgi**, then a space, followed by the full path to the CGI program itself (and any program parameters it might require). If the named program does not include a full path, GwebS assumes that the CGI program resides in the same directory as the one containing the `gwebs` executable.

Whenever GwebS receives a request starting with the Virtual-part, it starts the program named in Physical-path (including any parameters), sets up the environment for the program as defined in the CGI standard, sets the standard input to the CGI program to the TCP/IP data stream from the browser, and sends the program's standard output to the browser.

Note that you can also define CGI programs by extension. If you define a special Content-type named:

```
Content-Type    script/cgi                      cgi
Content-Type    script/cgi                      exe
```

all requested files with the given file name extensions are treated as CGI programs e.g.

```
Alias /utility       c:\gar\utilities
Content-Type         script/cgi      exe
```

A request `http://www.mydomain.com/utility/myprog.exe` results in GwebS starting `c:\gar\utilities\myprog.exe` as a CGI program.

Mapping a request to the Gweb status screen

```
Alias /server-status server-status
```

The Virtual-part represents the apparent location of the status screen. You could for example use something such as `/server-status.`

The Physical-path is the keyword **server-status**. Whenever GwebS receives a request that begins with `/server-status`, control is transferred to the Gweb status screen.

Mapping a request to the Web8/TDS-Web subsystems

```
Alias /ph14.gartest     web8  browser
```

The Virtual-part represents the apparent location of Web8 or TDS-Web. The Physical-path is the keyword **web8** or **TDS-Web**, followed by a string used as a session name.

If no session name is given, Gweb assumes a session name equal to "`node.application`" (see below).

The Virtual-part syntax is always

```
/node.application
```

where the node and application is information that your GCOS8 or GCOS7 administrator must give you.

Whenever GwebS receives a request that begins with `/node.application`, control is transferred to the Web8 or TDS-Web subsystem handler.

Password Virtual-part username:password Realm

```
Password /server-status admin:secret Admin Credentials
Password /admin-area    admin:secret Admin Credentials
```

The **Virtual-part** represents the web directory and underlying documents that you want to protect with a username and password combination. The **username:password** text is a username and password pair (delimited by a single colon) that protects the Virtual-part in question. The **Realm** is the name given to the protected area, and the text is used as a prompt displayed to users who browse to a location within the protected realm. The prompt is only displayed the first time; the user name and password are remembered. Different areas can be given the same realm name (prompt), and once the user name and password pair is given for one, it allows access to all areas in the same realm.

Note that neither the username nor the password can contain spaces. There can be multiple password settings for the same Virtual-part, but only the first Realm prompt defined will be used.

SSL Directives (GwebSS only)

```
SSLCertificateFile (required)
SSLCertificateKeyFile
SSLCertificateKeyPass
SSLCaCertificateFile
SSLCaCertificatePath
SSLVerifyClient
SSLVerifyDepth
SSLSessionCacheTimeout
SSLOptions
```

These directives can only be used in `gwebs.XXX` if GwebSS is to be used.

SSL parameters are described in separate document entitled *Using SSL for security in G&R products, and Stunnel as a Glink Telnet server* and will not be described further here.

Launching GwebS

When you have created the configuration file, Gweb can be launched with the command:

```
gwebs [-mi XXX]
   [-nb [off]|-na n.n.n.n ...
      [-nat m.m.m.m][-myip m.m.m.m]]
   [-dbg|-trace N] [-lp N] [-ci N]
   [-h] [-q] [-pid PIDFILENAME] (Unix only)
```

The `-nb` or `-na n.n.n.n` instruct the Gweb server to report statistics and load balancing information to Gproxy. With `–nb` (default setting), load balancing information will be broadcasted to all Gproxy systems on the current subnet. With `-na`, this information will be sent to Gproxy on one or more IP addresses. Up to four IP addresses can be supplied, separated by a space.

The `-nat m.m.m.m` parameter defines the IP address of Gweb to be given to clients connecting via Gproxy. This parameter is only needed when GwebS and Gproxy are located on a system on the protected side of a router doing Network Address Translation (NAT), and supplies the IP address that should be visible to client systems on the outside. Please consult the Gproxy manual for details on load balancing Gweb servers.

The `-myip m.m.m.m` parameter defines the IP address of the Gweb server itself, and is only used if one of the Gproxy parameters is set. Normally, GwebS is able to find out its own IP address, but in some circumstances, especially where the machine is equipped with multiple physical or logical network interfaces, the correct IP address must be set with this parameter. Please consult the Gproxy manual for details.

The `-trace N` parameter defines a trace level for all sessions started via this server. The `-dbg` parameter is equivalent to `-trace 9`.

The `-lp N` parameter defines the TCP/IP port number on which GwebS should listen. The default port number is 80. This overrides any port setting in the GwebS configuration file (see below).

The `-ci N` parameter defines the check interval in seconds (default 30). Each check interval GwebS will report its status to the server database file `_active.srv` that is located in the Host Links `servers` directory. This database can be viewed using Gmanager (See the Host Links Installation & configuration manual).

The last parameters are for the UNIX/Linux version only: `-p` prints a help screen, `-q` starts GwebS in Quiet mode (nothing is displayed on stdout), and `-pid PIDFILENAME` causes the parent pid of the process to be written in the file PIDFILENAME. This makes it easier to terminate the server, with for example the command "`kill $(cat PIDFILENAME)`".

On Windows servers, a dialog box will appear (unless the program is run minimized or as a service) where some statistics are updated, and there is a button for terminating the server. On UNIX/Linux, the program will make itself a background process. In both cases, you can shut down GwebS using the G&R product Gman (UNIX/Linux) or Gmanager (Windows).

GwebS will write to a log file and (depending on configuration) an access log file. These files are stored in the server's directory, which is

UNIX location:	`/usr/gar/servers/<scid>.gwb/`
Windows NT location:	`C:\gar\servers\<scid>.gwb\`

(<scid> = your DSA node name)

Launching GwebSS

The SSL version of the Gweb server uses the same parameters as the standard edition, with some additions.

```
gwebss [standard GwebS parameters ...]
   [-sslcf  | -SSLCertificateFile] (required)
   [-sslckf | -SSLCertificateKeyFile]
   [-sslckp | -SSLCertificateKeyPass]
   [-sslca  | -SSLCaCertificateFile]
   [-sslcap | -SSLCaCertificatePath]
   [-sslvc  | -SSLVerifyClient]
   [-sslvd  | -SSLVerifyDepth]
   [-sslto  | -SSLSessionCacheTimeout]
   [-sslopt | -SSLOptions]
```

These parameters have the same names as SSL configuration file directives, listed earlier. Command-line parameters override the corresponding configuration file parameters.

SSL parameters are described in separate document entitled *Using SSL for security in G&R products, and Stunnel as a Glink Telnet server* and will not be described further here.

GwebS performance tuning

There are some extra parameters that control the performance of GwebS. In normal situations you should not need to adjust any of these values, but in some circumstances you might need to define more explicitly how GwebS should operate.

These parameters can be defined both in the `gwebs.def` configuration file and on the command line. Command line parameters override configuration file parameters.

Cmdline	Config file	Default	Description
`-minidle`	MinIdleServers	4	The minimum number of idle server tasks (threads) waiting for a connection. GwebS creates new server tasks if there are too few idle, in order to be prepared for new connections.
`-maxidle`	MaxIdleServers	10	The maximum number of idle server tasks (threads) waiting for a connection. GwebS terminates server tasks if there are too many idle.
`-minrun`	MinRunningServers	4	The absolute minimum number of server tasks (threads). GwebS never uses fewer tasks than this.
`-maxrun`	MaxRunningServers	200	The absolute maximum number of server tasks (threads). GwebS never uses more tasks than this.

Cmdline	Config file	Default	Description
`-minidle`	MinIdleServers	4	The minimum number of idle server tasks (threads) waiting for a connection. GwebS creates new server tasks if there are too few idle, in order to be prepared for new connections.
`-maxreq`	MaxRequestsPerChild	0	The maximum number of browser requests a given server task handles before termination. The default of zero means that the server task never terminates. If you define a number, the server task terminates after handling that number. GwebS creates new tasks as necessary, depending on the parameters above.

Use the server-status mechanism described above to see details of GwebS server task usage.

Gweb configuration files

Gweb retrieves information on how to connect to mainframes from these files:

- `gweb.cfg` (see page 31)
- HTML start-up page (see page 70)

The `gweb.cfg` file should be used for all Gweb application parameters. The `cpic.cfg` file previously used for connection parameters is only supported for backward compatibility. HTML start-up pages show connection menus that allow the users to select specific mainframe connections and deliver parameters such as login information needed to make the connection. They can also be used to pick up connection information from the user interactively.

gweb.cfg

```
Default
   default parameters
host <host name>
 [-gweb]
   gweb parameters...
 -user
   emulation parameters...
 -li xxx
   line parameters...
```

The `gweb.cfg` file is located in `/config/default`, inside your Host Links system directory. It is divided into sections, one for default settings, and one section for each mainframe connection that to Gweb is able to establish.

The default section is preceded by the word `Default` on a single line.

```
Default
```

A single line with the following format precedes the settings for individual mainframe sessions:

```
host <hostname> -emu <emulation> -title "title"
```

where `<hostname>` is the name of the mainframe session. The optional `-emu` specifies one of (`dku`, `7800`, `3270`, `5250`) and `-title` can be any descriptive text. If `-emu` is used, then Gweb will be able to include the entry in the list of available sessions obtained if the user connects to Gweb without giving any parameters, e.g.:

```
http://www.gar.no/cgi-bin/gweb
```

The default heading and each of the host headings can be followed by any number of parameters.

By default parameters in a `host` section are Gweb parameters. The `-gweb` flag is only needed if you need to switch back from line or user parameters. All parameters after and including an `-li XXX` directive are regarded as line parameters; and all parameters after and including a `-user` directive are regarded as emulation parameters.

A Gweb delivery includes a sample `gweb.cfg` file that must be edited to meet your own requirements. A sample of this file is:

```
Default

  -htmlhead gwebhead.htm
  -htmlfoot gwebfoot.htm
  -prt  gwebprt.htm

host tp8v2 -emu 7800 -title "TP8 on System J (7800 mode)"
 -gweb
  -fkeys 0
 -li dsa:ggate.gar.no
  -co tp8v2
 -user
  -snd off
  -rw  10
```

Gweb parameters (-gweb)

HTML/HTTP options

Gweb generates HTML pages at level 4.01 (Strict). Browsers must support this. Gweb uses CSS (Cascading Style Sheets) to control the appearance of the generated pages, and assumes that the browsers support CSS 2.1. The parameters previously used to control HTML colour rendition and set a CSS level are obsolete.

HTML/HTTP options	
`-acc #rrggbb` OBSOLETE, browsers must support CSS 2.1	'`-acc`' is a code that identifies a colour and intensity, e.g. `-at` identifies turquoise, `-atl` identifies low intensity turquoise. `#rrggbb` defines the HTML color code to use for the colour. HTML color codes are specified in the form `#rrggbb`, where `rr`, `gg`, and `bb` are hexadecimal values in the range `00` to `FF` which define the red, green, and blue component of the color, respectively.
`-css <level/1>` OBSOLETE, browsers must support CSS 2.1	Specifies the level of support for Cascading Style Sheets (CSS). The default is one, which enables support for all CSS level-1 elements. `-css 0` disables all usage of CSS extensions to HTML, while `-css 2` instructs Gweb to use CSS up to level 2. If not disabled, Gweb uses CSS to define a monospaced font and the text color in input fields.
`sssss` `(see below)`	Specifies HTML files, used in the parameters below. Gweb looks for these files in the `/html/gweb` directory, inside your Host Links system directory (i.e. the same directory as this documentation). References to other files or images made in these HTML files should point to files in the normal document tree of your web server.

HTML/HTTP options	
`-htmlhead` **`sssss`**	Specifies an HTML file that Gweb will insert as a prefix to all of its HTML pages. This file would usually specify a title and a background image, plus possibly display a company logo. An example prefix file might look like this: `<html>` `<head>` `<title>My Gweb pages</title>` `</head>` `<body` `background="images/mybackground.gif"` `>` `<img src="images/mylogo.gif"` `alt="Company Logo">` A more detailed prefix file is in `gwebhead.htm`. The prefix and postfix (see below) files are looked for in the following order, if an absolute path is not specified: 1. In `/usr/gar/html/gweb/`**`[sessionname]`** 2. In `/usr/gar/html/gweb`
`-htmlfoot` **`sssss`**	Specifies an HTML file that Gweb will insert at the end of its HTML pages. This file could include a Webmaster's e-mail address or similar information. An example postfix file might look like this: `<a href=` `"mailto:webmaster@mycompany.com">Webmaster</a>` `</body></html>` A more detailed postfix file is in `gwebfoot.htm`
`-httpresp` **`xxxx`**	Defines a HTTP Response header line that is sent to the browser as part of the response. For example, to define for proxies and browsers that facelifted Gweb HTML pages never should be cached, add the parameter `-gweb` `-httpresp "Pragma: no-cache"` to the configuration, and that line will be included in the HTTP response header. You may supply up to 10 different HTTP response header parameters.

HTML/HTTP options	
`-session-cookies` **`OFF`**`/on`	Specifies that a session cookie will be sent as part of the HTTP response. The cookie contains enough data to identify the session for the browser that initiated it, and expires when the session expires or is terminated.
`-title` **`xxx`**	Page title for all HTML pages. By default, the page title is the session name.
`-wmlhead` **`sssss`**	Specifies a header file that Gweb will use when a WML (WAP) output format is generated. If not specified, Gweb uses the following: `<?xml version="1.0"?> <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml"> <wml> <card id="GwebForm" title="#gweb(title)#"><p>`
`-xmlhead` **`sssss`**	Specifies a header file that Gweb will use when a XML output format is generated. If not specified, Gweb uses the following: `<?xml version="1.0" standalone="no"?> <!DOCTYPE hostScreen PUBLIC "-//GAR//DTD Gweb XML V6.3.0//EN" "http://#gweb(server)#/gweb/gweb.dtd">`

Facelifting options

Facelifting options	
`-cache N/0`	Defines the maximum number of facelifting files kept in the memory cache. The default value is zero, which means that the memory cache is disabled. Any other value means that the contents of the **N** first files are stored in the cache. The file cache is used in order to avoid file I/O. The first time a file is used, for example the page footer file (`htmlfoot.htm` or similar) it is loaded into memory, and stays there until the session terminates. Subsequent accesses pick up the contents already saved in memory, avoiding disk/file I/O. The `#dump cache#` macro can be used to display simple statistics of the file cache.
`-capture OFF/on`	Enables capturing of all host forms in a session. A pair of files (`nnnnnnnn.gcf` and `nnnnnnnn.htm`) are saved in the /usr/gar/html/gweb/SESSIONNAME/ directory. Starting at 00000001, the number is increased for each new form received from the mainframe. This feature is most often used when performing extended forms facelifting, where the captured files serve as input for the Gwebedit facelifting editor.
`-capture1 OFF/on`	The `-capture1` option lets you decide which host forms to capture, by displaying a button labeled "**Capture**" together with the host form data. Pressing that button causes that single host form to be captured to disk.
`-capture js`	Enables the JavaScript method for forms capture. The current host form appears in an extra browser window. You identify the form by marking one or more areas of text with the mouse. The form is saved into the index.cfg facelifting database after you have supplied the logical form name.

Facelifting options	
`-htmldir` **`sssss`**	Defines the default directory to search for facelifted versions of host forms. The default value is **`#S/html/gweb`** where **`#S`** is your System Directory (usually `c:\gar` on Windows, and `/usr/gar` on UNIX/Linux systems).
`-param`**`N`** `xxxx`	Defines a user-supplied parameter for this session, where N is an integer value from 1 to 50, and xxxx is the parameter value. See page 128 for details. Example: `-gweb` `  -param13 PARAMETER NO. 13` `  -param50 HELLO`
`-stack N/`**`0`**	Defines the maximum number of host forms kept in the form stack. The default value is zero, which means that the form stack is disabled. See page 117 for details.

Screen presentation options

Screen presentation options	
`-afl _`	(Automatic Field Length) If Gweb receives a screen from the host where no variable fields are defined, it creates (simulates) an input field starting at the current cursor position. By default, the width of this variable field extends to the end of the line. This screen field can be shortened automatically, depending on the rules below: 1. If there is a change in the visual attributes (for example, from Inverse Video to Normal) before the end of the line, the point where the attribute changed is considered the end of the field. 2. If the field starts with one of the following characters: **underline** (_), **period** (.) or **space** ( ); then the width of the input field exactly as long as the number of these characters. 3. The `-afl X` option specifies which character is used for field "padding". If the **X** character (default value is the underline (_) character) exists within the field, the created field is no longer than the last **X** character. The field width will never extend beyond end of line. See also the `-cf` option above. To disable the logic above, supply the parameter `-afl 0` (zero).
`-cf` **OFF**`/on`	If Gweb receives a screen from the host where no variable fields are defined, it creates (simulates) an input field starting at the current cursor position and extending to the end of the line (but see the `-afl on` option below). By default, any content on the screen is retained inside this variable field. The `-cf on` option specifies that this field should be created empty.

Screen presentation options	
`-fp <option>`	Applies field padding. The different options are described below: **`none`** Disables field padding **`underline`** Enables 'underscore' padding. This has two consequences: ▪ Trailing underscore characters are removed from the input stream (i.e. from the screen form sent from the host) before the data is presented in the browser. ▪ Underscore characters are appended to each field that is not completely filled, if the field was padded with underscores by the host. **`space`** Enables 'space' padding: Space characters are appended to each field that are not completely filled, before the fields are sent to the host. The default is that no padding is to be performed.
`-fs OFF/on`	("Full screen" mode) Defines that the host screen in automatic facelifting mode will be displayed in a border, guaranteeing a 24x80 image that more closely resembles the image on the physical terminal.

Screen presentation options	
`-httpcset` **xxxx**	Defines the character set used in the output HTML pages. If not specified, the character set is assumed to be ISO-8859-1, which is the HTTP default if no character set is specified. If specified, the **xxxx** character set specified here will be added to the `Content-Type` HTTP response header, instructing your browser to use the specified character set when displaying the page. For example, to display Greek characters you would have this parameter in the gweb.cfg file: `-gweb` `  -httpcset ISO-8859-7` and this would cause Gweb to return `Content-Type=text/html; Charset=ISO-8859-7` in the HTTP section of the response.
`-semigraphics` **ON**`/off`	Gweb can represent semi-graphic characters using Unicode or approximation by display characters. If neither presentation is pleasing on your browser, you can turn off semi-graphics completely, and the screen positions will be filled with space characters.
`-unicode` **ON**`/off`	By default, Gweb will use characters from the extended set defined by Unicode if such characters are needed to represent semi-graphic symbols, and if the browser supports Unicode. You can suppress Unicode semi-graphics with the directive `"-unicode off"`, in which case semi-graphics will be approximated with display characters.

Screen presentation methods	
`-pm method`	**Presentation Method:** Depending on the application, need for alignment and available bandwidth, Gweb's automatic facelifting can use different presentation methods.
`-pm 0` `-pm SIMPLE`	**Simple presentation:** The screen content is displayed inside <PRE> HTML tags. Variable fields are defined at the position they occur. This can lead to columnar misalignment, because the different browsers usually place a thin 3D border around input fields and slightly adjust the font size, causing the width of the input field to be different from what it would be had it been a fixed field. This is the presentation method used in earlier releases of Gweb.
`-pm 1` `-pm TABLE`	**Tabular presentation:** This presentation method uses a <TABLE> mechanism to display the screen contents. This ensures that columnar alignment is improved, although this requires somewhat larger HTML output. It can also lead to odd word spacing in some circumstances. This is the default presentation method.
`-pm 2` `-pm CSS`	**CSS-based presentation:** This presentation method uses fine-tuned CSS to ensure perfect columnar alignment. The CSS settings vary from browser to browser, and focus on 4:3 aspect ratio screens. Wide screens are not fully supported.
`-pm XML`	**XML output:** This presentation method causes Gweb to output host screen information in XML format. This is not a display format intended for browsers, but a data transfer format for applications that understand XML. XML developers may check the Gweb DTD on the file gweb.dtd for details.
`-pm WML`	**WML output:** This presentation method causes Gweb to output host screen information in WML format. This is a display format intended for WAP browsers, most commonly found in modern cellular phones.

JavaScript settings

These parameters define the JavaScript calls that Gweb adds to the HTML code used to define HTML form elements when generating automatically facelifted screens.

As delivered, Gweb includes JavaScript code `gweb.js` that implements these calls. The supplied code uses data inserted into the HTML form by the `#js(variables)#` macro. The JavaScript routines perform functions such as cursor positioning, field validation, attribute checking, automatic tabulation, Enter=Transmit and so on that make the browser keyboard and HTML form function more like a terminal keyboard and screen with variable fields.

You can change these standard JavaScript settings if you write your own JavaScript to handle your calls. The entire parameter string must be on a single line in `gweb.cfg`, and enclosed in single quotes.

To suppress the JavaScript code, supply these parameters without any value. For details, check the JavaScript Automation chapter on page 135.

You can use Gweb facelifting macros to insert these same calls into your own customized HTML files when facelifting screens.

JavaScript settings	
`-jsfield 'XXX'`	For variable fields within a form - i.e. within the `<INPUT TYPE=text NAME=xx ...>` HTML tag: **`'onKeyPress="return doKeyPress(event);" onKeyUp="return doKeyUp(event);" onFocus="doSelect(this);"'`** The contents of this parameter is available to facelifted HTML pages with the macro `#js(Field)#`.
`-jsform 'XXX'`	For form definitions - i.e. within the `<FORM METHOD=POST ACTION="#gweb#" ...>` HTML tag: **`'onSubmit="doForm(event);"'`** The contents of this parameter is available to facelifted HTML pages with the macro `#js(Form)#`.

JavaScript settings	
-jsbody 'XXX'	For HTML body definitions - i.e. within the <BODY ...> HTML tag: **'onLoad="doFocus()" onKeyDown="return doKeyDown(event);"'** The contents of this parameter is available to facelifted HTML pages with the macro #js(Body)#.
-jsbutton 'XXX'	For submission buttons - i.e. within the <INPUT TYPE=SUBMIT ...> HTML tag: **'onFocus="doSelect(this);"'** The contents of this parameter is available to facelifted HTML pages with the macro #js(Button)#.
-jshref 'XXX'	For hyperlinks - i.e. within the <A HREF="#gweb# ...> HTML tag: **'onClick="doClick(this);"'** The contents of this parameter are available to facelifted HTML pages with the macro #js(Href)#.

Host printing settings

Host printing settings	
-prt sssss	Specifies an HTML file that Gweb will use when a print is requested. If not specified, Gweb uses the following: `<html>` `<head><title>#title#` `print</title></head>` `<body><pre>` `#print text#` `</pre></body></html>` We deliver gwebprt.htm in the Gweb directory. This includes JavaScript to print the data, and close the print window. See the section entitled ***Print Support*** for details.

Button and F-key parameters

Button and F-key parameters	
`-bp t/B`	Button Position — tells Gweb to put buttons on the top or bottom of the page. The default is bottom.
`-brk ON/off`	Displays/hides the Break button. The default is on.
`-cp OFF| radio|auto`	Some host applications read and react on the cursor position. When this parameter is set to **radio** Gweb will place radio buttons next to all the input fields on its HTML screens to allow explicit cursor placement. When the parameter is set to **auto** the Gweb JavaScript automation routines will use the last variable field as the host cursor position. The default values are **auto** for the IBM3270 and IBM5250 mode, and **off** otherwise.
`-disc ON/off`	Displays/hides the Disconnect button. The default is on.
`-fkeys n`	Defines the maximum number of FKEY buttons visible; 0 shows none, 5 shows F1-F5, etc. The default is 24 for the IBM emulations, 12 for others. See `-fkon` for values of `n` and the corresponding function keys. Note that `-fkeys <N>` overrides all previous `-fkon` or `-fkoff` settings, but any `-fkon` or `-fkoff` parameters that follow `-fkeys` will adjust it. So `-fkeys 5` `-fkoff 3` `-fkon 26` instructs Gweb to show F1, F2, F4, F5, and PA2.

Button and F-key parameters	
`-fkon <n>`, `-fkoff <n>`	Explicitly directs Gweb to display/hide the given Button or Function Key. Possible values for `n`, the function key and the `text` displayed on the button are: 1-12 VIP and DKU emulations: Function keys `F1` thru `F12`. IBM emulations: Function keys `PF1` thru `PF12` 13-24 VIP and DKU emulations: Function keys `SF1` thru `SF12` (F1-F12, shifted). IBM emulations: Function keys `PF13` thru `PF24` 25-27 IBM 3270: Function keys `PA1` thru `PA3` 32, 36, 37 The `Transmit`, `Break` and `Disconnect` buttons. Break has text `ATTN` for IBM. 34 The `Refresh` button. Gweb requests the emulation to refresh the screen without any application interaction. Useful in situations where application screen or print data continues to arrive after Gweb has displayed the screen. The button updates the screen, or receives the print that has arrived afterwards without disturbing the application. 38 IBM emulations: `SysReq` 39-45 IBM 5250 buttons: `TEST`, `RollUp` and `RollDown`, Help, `Field+`, `Field-`, `FieldExit`. 46 VIP emulation: `Return`
`-fn <`**`CPIC-KEYCODES`**`>`	User-defined macros on function keys. See `-fkon` for values of `n` and corresponding keys. `CPIC-KEYCODES` are keystrokes given to the emulation when you press the function key. Check the Appendix on page 141 for details.

Connection and communication parameters

Connection and communication parameters	
`-closewindow on/`**`OFF`**	Causes the browser window to be closed automatically if the mainframe session is terminated normally.
`-emu emulation`	Selects the emulation to be used. One of `dku`, `7800`, `3270`, `5250`. Valid only in the `host` heading for a connection. If used Gweb will include the connection in the list of available connections.
`-guard` **`ON`**`/off`	If the user attempts to close the web page or browse to another web page without closing the mainframe connection, the guard gives a warning.
`-stay` **`OFF`**`/on`	If set, when the user navigates to a host screen not already known by Gweb (i.e. not facelifted), the host session is immediately disconnected. This feature can be used in situations where only some of the application screens are facelifted, and the security policy prevents the user's access to other host screens.
`-hc kkkkkk`	If `-snd` is **on** (i.e. Gweb starts in the send state) Gweb will transmit this string of keystrokes immediately after connecting instead of giving the user a command line prompt. This can also be set in the connection form, see the `hc` field description below. If `-snd` is **off** (i.e. Gweb starts in the receive state) Gweb will transmit this string of keystrokes immediately after the first host screen is received, and will not send any data to the browser until the next host screen (number 2) is received. This feature can, for example, be used as an automatic log-on macro.
`-hd kkkkkk`	Gweb transmits this string of CPIC keystrokes immediately before disconnecting from the host. This feature can, for example, be used as an automatic log-out macro.

Connection and communication parameters	
`-maxpages n/0`	This parameter defines the maximum number of output HTML pages Gweb can send to the browser. If this parameter is specified, Gweb will automatically disconnect the session after the N'th output page has been sent. This feature allows you to create facelift scenarios where the one and only response from Gweb contains all the information the user needs, and eliminates the need for a short timeout or a Disconnect button. The default value of zero disables page counting.
`-rs nn/5`	This specifies a refresh speed, default 5 seconds. It is used if the application keeps the initiative longer than the time-out specified with the emulation `-dw` or `-rw` parameter. The screen is updated with data received so far, and then the browser automatically reconnects every 5 seconds to refresh the display.
`-snd on/OFF`	Normally Gweb assumes that the mainframe will begin the dialog after connection, and will wait for a banner from the application. If there is a host command (`-hc`) configured it will be sent when Gweb is given the initiative (turn) after the banner. For applications that expect the user to start the dialog 'send' must be on to give Gweb the initiative after connection. If a host command `-hc` has been configured or the connect form has field `hc` then it will be transmitted, otherwise Gweb will display a prompt for a user command.
`-timeout n/5`	Session timeout: This controls how many minutes Gweb keeps an idle session running before it is disconnected. You should increase this value for sessions where the users need more time between inputs. The default is 5. The `Timeout N` directive in the Gweb facelifting configuration file overrides this parameter.
`-title xxx`	Any descriptive text for a session. Valid in the `host` heading entry if `-emu` is also included. Overrides the session name as a session description.

Connection and communication parameters	
`-unlock` **`xx`** ***`xx`*** *`values:`* `co/dn/da/dx/ du/dp/db/lu/ mn/mx/pco/pw /res/ur/ti/ tm/all`	Defines which G&R/Gline parameters can be set by the user from the Gweb connection page. Normally the user cannot change any line parameters, and Gweb will only use those parameters specified in the configuration file. If you want individual users to be able to change line parameters — for instance, to enter their own user names and passwords — you have to unlock the necessary parameters. If users are allowed to change the project, username, and password settings for a given connection you would enter these three lines into the `gweb.cfg` file: `-unlock dp` `-unlock du` `-unlock pw` For information on how these parameters should be specified in the Gweb HTML connection form, see below.

Emulation parameters (-user)

This chapter contains an overview of the different emulation parameters that can be set in the `-user` section of the `gweb.cfg` configuration file.

Connection and dialog control

There are several issues concerning Gweb in relation to the browser and the mainframe application:

- Send (`-snd`), establishing the initiative (turn): Unfortunately Gweb is not able to rely on the results of the 'turn' negotiation, even for protocols where this is defined. Gweb therefore assumes that the mainframe application has the initiative, and will wait for the application to send a welcome banner. If this is not the case, you must configure Gweb to have the turn (`-snd on`). Gweb will then send the configured host command (if any) or display a prompt on the browser so that you can enter an initial command. Note that (`-snd`) is a Gweb parameter, not a user parameter.

- Blank screens (`-bs`), avoiding blank screens: Unfortunately some of the gateways send multiple 'turn' signals. Often a false turn signal is received before the application has sent any data at all, leaving the user stranded with a blank screen. Gweb attempts to avoid this by assuming that if the mainframe application has sent no data at all to the screen, then any turn signal is false. If your application genuinely sends a blank screen, giving you the initiative, then you must configure Gweb to allow blank screens (`-bs on`). Note that this is always the case for Bull transactional systems that are configured to give the user the initiative on connect, and which expect a 'logon' command. These must always be configured to allow blank screens, as well as giving Gweb the initiative after connect (`-snd on`).

- Additional refresh wait (`-arw`), waiting for a genuine turn signal: If the 'blank screen' protection is not enough or if you have to allow genuine blank screens, and your gateway is sending premature turn signals leaving you stranded with incomplete output on the screen, then you must configure Gweb to wait for some period after a turn signal, to see if the application is genuinely finished, or if the turn signal was false, and more data follows. The `-arw` parameter allows you to specify a number of tenths of a second that Gweb will wait after a turn, in case more data follows.

- Delay turn (`-ddt`), delaying turn at the line level. Sometimes the mainframe delivers data and the turn, but then demands that the line module give back

the turn. By setting this **line handler parameter** (after `-li`) to e.g. 5 tenths of a second the turn signal is delayed, and cancelled altogether if the mainframe demands the turn back, preventing premature action by Gweb. This can avoid the need for `-arw` described above.

- Data wait (`-dw`), limiting wait time for output data: After you transmit input Gweb must wait for application output. If output is received, but not the turn, Gweb will wait again for more (but see 'Refresh wait' below). During these 'data waits' the session is completely blocked, and you are not able to take action from the browser. You can limit this wait time using `-dw` to specify the maximum number of seconds that Gweb will wait for data. If you consider that the application should never take more than ten seconds to deliver at least a part of the data, then setting `-dw 10` would ensure that in case of a longer wait, control will be returned to Gweb. Gweb will display any data received so far, and give you the opportunity to take action. Gweb will also instruct your browser to reconnect periodically to pick up any new data. If your mainframe application offers you the opportunity of starting a job, and does not send any output at all until the job is completed, then you might want to set the data wait time short, so that you get a timeout, are given control in the browser and can decide to let the automatic refresh mechanism show you the mainframe screen periodically (see the Gweb `-rs` parameter) or take other action (Break).

- Refresh wait (`-rw`), limiting wait time for turn: It may be that the application is returning data fast enough to avoid the 'data wait' timer, but that many output blocks are sent without the turn. This is always the case with some mainframe monitor systems that continuously update a display. In these and other cases where mainframe data is being sent often enough to avoid the data wait timer, but is taking an unreasonable amount of time to send the turn, Gweb must be configured to limit the time it will wait. You can limit this time by using the `-rw` parameter to specify the number of seconds you are prepared to wait before you wish to receive the data that has arrived so far, and be given the opportunity to take action. Gweb will also instruct your browser to reconnect periodically to pick up any new data (see the Gweb `-rs` parameter).

Almost all Gweb connection and dialog problems are related to the issues discussed above, and can be avoided by careful usage of the related parameters.

Note that when connecting to Bull GCOS systems most problems caused by false turn are due to TNVIP or other terminal gateways. These problems can be avoided, and response times improved if you connect to GCOS via *G&R/Ggate*.

Connection and dialog parameter overview

These are in all the emulations, and should be used in the `-user` section of the `gweb.cfg` configuration file. Please read the ***Connection and dialog control*** discussion above before using these parameters.

Parameter	Description
`-ARW NN/0`	'Additional Refresh Wait' deals with situations where Gweb prematurely receives a TURN signal. Gweb waits `NN` tenths of a second for more data after it has received TURN, before it sends the current screen to the browser. But see the `–ddt` line handler parameter.
`-BS on/OFF`	'Blank screen' specifies whether the emulation will return control to Gweb without receiving any data. The default is not to do so, the application is expected to respond when Gweb connects or transmits. This must be set on for applications that genuinely send blank screens, and wait for input, particularly Bull systems configured to start with a 'logon' command.
`-DW NN/10`	'Data Wait' specifies how many seconds Gweb blocks the user, waiting for application data. It applies both for the initial wait for any data at all, and then for subsequent waits for more data (see `-RW`). If this timer expires, the current screen is delivered to the browser, and the browser actions are as described in the ***Connection and dialog control*** discussion above.
`-RW NN/20`	'Refresh Wait' specifies the number of seconds Gweb waits after each send for the application to return the initiative (turn). The time-out is checked each time the application returns data, and if the turn is kept by the application, and the timeout has not expired, a new 'Data Wait' period is started (see `-DW`). If the timeout expires, the current screen is delivered to the browser, and the browser actions are as described in the ***Connection and dialog control*** discussion above.

VIP7800 parameters

VIP7800 emulation parameters should be set in the `–user` section of the `gweb.cfg` file: defaults in upper case.

Parameter	Description
`-ACL on/OFF`	Add `CRLF` to the end of lines in text mode, even if `-SS` (space suppress) is set.
`-AL on/OFF`	Add `LF` to incoming messages terminated with CR.
`-AM /1/2`	7200 attributes, 0=No, 1=Yes, 2=Extended. This option specifies whether the emulator should accept the 7200 set high intensity and set low intensity commands. This should not be necessary unless you are using an application written specifically for the 7200 terminal. Note that you may not mix 7200-type and 7800-type attributes on the same screen under any circumstances. An additional setting, 'extended', allows use of two extra commands for invisible and blinking fields that were supplied with some 7200 hardware emulations. The default is 0, no 7200 attributes accepted.
`-ATTX YYY`	Attribute mapping. The VIP attribute `X` will be mapped to `YYY`, where `Y` is a V78sim visual attribute. See V78sim manual, section entitled *Attribute mapping*.
`-DBG on/OFF`	Turns on an emulation level trace.
`-DBS on/OFF`	`BACKSPACE` erases character to the left of cursor.
`-EC on/OFF`	Use SS2 to send 8-bit characters on a 7-bit line. This is a private arrangement between V78sim and G&R UNIX applications. See -UI for standard encoding.
`-E8 on/OFF`	Use the extended ASCII value (eight bit) for real national characters when sending to the host.
`-EDO value`	End data only mode. Decimal value identifies the character that will end data only mode. See `-SDO`.
`-ES on/OFF`	Extended status. When the option is turned on, Gweb 7800 will send additional bytes when a remote inquiry is received.
`-ETX ON/off`	Send ETX as message terminator. When turned off, EOT is sent as message terminator.

Parameter		Description
-IB	on/OFF	Initial setup of Block mode. The parameter has effect only when Gweb 7800 is started in text mode, and will normally be set by the host at connect time. DO NOT turn it on unless you are absolutely sure that you need it – you will otherwise experience apparent 'hangs' when transmitting large blocks of data to the host in text or forms mode. When turned on, the initial block size is 256 bytes.
-IT	on/OFF	Initial setup of text mode or character mode. The default is character mode, but when used with the DIWS, DSA or TNVIP line handler V78sim will force text mode as the initial setup.
-KUW	on/OFF	Wait for keyboard unlock. Causes the emulation to regard the VIP7800 keyboard-unlock control sequence as a turn signal. The 'turn' is lost when the emulation transmits, and regained when the keyboard is unlocked. This option excludes the -TSW option.
-LC	ON/off	Lower case is transmitted to the host. Can be turned off so that all data to the host is transmitted in upper case (although it is displayed in lower case on the screen).
-LK	ON/off	The keyboard is locked on transmit, and freed on 'turn'. For two way simultaneous sessions with no turn, the keyboard is freed on the first host reply. Macros are suspended from when they transmit until the keyboard is freed. For two-way simultaneous sessions with a host that does not reply this option must be turned off.
-PT	ON/off	Transparent print addressing in the VIP protocol header is standard, but can be turned off if the mainframe sends both print addressing in the VIP header and in the data stream itself, enclosing the print data between start- and stop- print control sequences (TNVIP does this).
-R72	on/OFF	Enables VIP7800 72 line mode. The default is 24 lines.
-RPD	on/OFF	Remove DEL and NUL characters addressed to the printer.
-SCR	on/OFF	Host overflow is handled by scrolling upwards, instead of waiting for CR or wrapping (-WR).
-SDO	value	Start data only mode. Decimal value identifies the character that will put V78sim into data only mode. This turns off the

Parameter		Description
		emulation, and delivers the mainframe data directly to the user terminal, which is usually a screen-scraping application. See `-EDO`.
`-SF`	`on/OFF`	This is a special option to use fields in the order they are defined rather that in the order they appear on the screen. Don't use this unless you have an application that has this particular need.
`-SFF`	`on/OFF`	Skip all Form Feeds in printer data.
`-SSP`	`ON/off`	Space suppression. If set, on transmission trailing spaces at the end of lines and/or fields are removed. In text mode each line is terminated with CRLF. In forms mode each field where spaces are removed is terminated with HT. Fields with no trailing spaces are terminated with US. If not set, all lines/fields are filled with trailing spaces and sent as a continuous stream with no delimiters. But see `-ACL`.
`-TSW`	`string`	Specifies a string in hex (HxHxHx e.g. 1B5B etc) that will be regarded as the turn signal. The 'turn' is lost when the emulation transmits, and is regarded as regained when the string arrives. This option excludes `-KUW`.
`-UI`	`on/OFF`	Use SISO algorithm to send real national characters (eight bit characters) to the host.
`-XL`	`US`	Translation from *`Host Links`* (ISO/Do11) 8-bit characters to 7-bit equivalents to the host, and vice versa. The correct `–XL` (`GB, GE, FR, SF, DE, NO, SP, IT, JA`) must be specified if you choose an 8 bit profile in the `profiles` file and communicate with a 7-bit national host.
`-XX`	`hxhx`	Any incoming character from the host can be translated into any other for display purposes. Both are expressed in hex, and the first becomes the second.

DKU parameters

DKU emulation parameters should be set in the `-user` section of the `gweb.cfg` file: defaults in upper case.

Parameter		Description
`-An`	`rgbkilu`	Override attributes 1-8; see ***colour and attributes***.
`-Ax`	`rgbkilu`	Set attributes to be used in fields; see ***colour/attributes***.
`-AL`	`on/OFF`	Do not release printer (host print) for sharing with others.
`-AS`	`on/OFF`	Simulate original VIP7700 (see `-VP` for Questar VIP7700).
`-BB`	`on/OFF`	Blink/blank attributes set with circumflex and tilde.
`-BV`	`on/OFF`	If Blink/Blank, show the circumflex and tilde.
`-CC`	`on/OFF`	Invalid Esc sequences give a warning message.
`-CL`	`7G`	Choose colour mode (`4A/4B/7Q/1M`) for colour rendition. The default interprets colour but leaves faint etc. The 7Q mode suppresses the faint, blink and underline like the DKU7211.
`-CS`	 `<NONE>` AR GR JA KO RU ZH ZH5	Character Set mapping and switching between different modes: single- or double-byte, or left- or right justified. The default is no character set mapping or switching. Arabic Greek Japanese (Shift-JIS) Korean (Hangeul) Russian (Cyrillic) Chinese (GB2312) Chinese (Big5)
`-CUD`	`on/OFF`	Cursor up/down in a form will move to the nearest field above/below, rather than the left-most/right-most field.
`-CUF`	`ON/off`	Allow cursor forward/back out of a variable field.
`-DBG`	`on/OFF`	Turns on an emulation level trace.

Parameter		Description
-E8	on/OFF	Exchange eight bit bytes with the line handler. The default is to map down to 7-bit on transmission and cut off the top bit on reception, but see -EC. Requires an 8-bit line (DSA direct or Ggate). Forced on by line parameter -tm dku9107.
-EC	ON/off	Codes/decodes SS2 to send/receive 8-bit characters on a 7-bit line. See -L2 for Latin-2.
-ET	on/OFF	ETX embedded in host data truncates the block.
-L2	on/OFF	Use the Latin-2 SS2 encoding table. Requires -EC.
-LC	ON/off	Lower case is transmitted to the host. Can be turned off so that all data to the host is transmitted in upper case (although it is displayed in lower case on the screen).
-LK	ON/off	The keyboard is locked on transmit, and freed on 'turn'. For two way simultaneous sessions with no turn, the keyboard is freed on the first host reply. Macros are suspended from when they transmit until the keyboard is freed. For two-way simultaneous sessions with a host that does not reply this option must be turned off.
-NL	ON/off	Generate new-line after transmits to the host in normal mode. This is standard, but it can be convenient to suppress it when working with some editors that generate a new line themselves and would get two lines with the standard mode.
-PL6	on/OFF	Scans print output interpreting or removing Bull print sequences, and interprets SS2 sequences as national (8bit) characters.
-PPOS	on/OFF	Physical positioning is not standard but is required when simulating some VIP7760 implementations.
-PRTID	24	Identifies the printer for the device attribute inquiry.
-PRNBC	80	Number of printer columns for device attribute inquiry.
-PRNBL	160	Number of printer lines for device attribute inquiry.
-PRCPS	80	Number of characters a second for device inquiry.

Parameter		Description
-PSTR	HxHxHx	Spool flag of up to 30 bytes expressed in hex. Causes the current print to be terminated. Any remaining data in the block goes to the next file for spooling later.
-PT	ON/off	Transparent print addressing in the VIP protocol header is standard, but can be turned off if the host sends both print addressing, and encloses Esc Z addressing as well. Suppressing the transparent print avoids the Esc Z being sent to the printer.
-QC	ON/off	Checks fields with Questar attributes numeric, pure numeric and signed numeric at the character level when typed. If turned off for efficiency, they will be checked at the end of the field. This should be turned off if most fields in the forms are numeric, but high efficiency is needed.
-QG	ON/off	The SI/SO characters are interpreted as start/end graphics. Some old applications use these for print addressing or other special purposes.
-QS	on/OFF	SDP-attribute rendition, normally set by host if needed. In this mode attributes do not take a space on the screen, and they are applied to the following display characters, not to the field.
-RJS	x	Right justification char. Default space.
-SCR	on/OFF	Host overflow is handled by scrolling upwards, instead of waiting for CR or wrapping (-WR).
-SFF	on/OFF	Interpret/filter printer Escape sequences, skipping all Form Feeds. Interprets SS2.
-TV	on/OFF	Host data overflowing variable fields is truncated instead of overwriting fixed data (Questar) or skipping to the following variable (original VIP, -AS).
-VP	on/OFF	OFF gives a Home after transmit, and variable fields end at FS. Needed for Questar-style VIP7760 emulation. ON gives a new-line after transmit and ends variable fields at the first display character after FS. Needed for Questar-style VIP7700 emulation. See -AS for a closer simulation of the original VIP7700.

Parameter		Description
-WCOM	on/OFF	Use Wincom definition of VIP numeric (GS 4).
-WR	on/OFF	Host data overflow is handled by wrapping, rather than waiting for CR, or scrolling (-SCR).
-WRB	on/OFF	Host data overflow is handled by wrapping as above, but the screen is blanked before output continues.
-WT	ON/off	The default is that tab beyond the end/beginning of the screen will wrap around to the beginning/end. The Questar standard is no wrap.
-XD	on/OFF	Sometimes the Datanet has been patched to translate all characters (for national character purposes), and you need to translate only the characters within DC3 Y X (cursor positioning) sequences back to their original values.
-XL	US	Translation from *Host Links* (ISO/Do11) 8-bit characters to 7-bit equivalents to the host, and vice versa. The correct -XL (GB, GE, FR, SF, DE, NO, SP, IT, JA) must be specified if you communicate with a 7-bit national host.
-XX	hxhx	Any incoming character from the host can be translated into any other for display purposes. Both are expressed in hex, and the first becomes the second. Normally used only on data characters, but can be used on DC3 sequences instead (-XD).

Colours and attributes

The Questar DKU7211 generates colour from a combination of other attributes. It has several modes: 4-colour A generates 4 colours by combinations of underline and low intensity. 4-colour B generates the same by combinations of blink and low intensity. It generates 7 colours by combinations of blink, underline and low intensity. The mode is set by parameter (`-CL`) or by the host. You can map any of these colours to any other, by changing the default for attributes 1-8 using the (`-An`) parameter as explained below.

Using terminal mode DKU7211 (`-TM DKU7211`) will cause many GCOS7 applications to generate screen images with colour attributes at some cost in efficiency, so you should otherwise use the default DKU7107 type.

Colour decoding (7, 4A, 4B)

Terminal	DKU Colour rendition table				Gweb DKU equivalents	
	Blink	**Underline**	**Low**	**Colour**	**Parameter**	**Default**
7 colour	No	No	No	White	`-A1`	`RGB`
	No	No	Yes	Turquoise	`-A2`	`BG`
	No	Yes	No	Green	`-A3`	`G`
	No	Yes	Yes	Red	`-A4`	`R`
	Yes	No	No	Yellow	`-A5`	`RG`
	Yes	No	Yes	Blue	`-A6`	`B`
	Yes	Yes	No	Violet	`-A7`	`RB`
	Yes	Yes	Yes	Black	`-A8`	
4 Colour A		No	No	White	`-A1`	`RGB`
		No	Yes	Turquoise	`-A2`	`BG`
		Yes	No	Green	`-A3`	`G`
		Yes	Yes	Red	`-A4`	`R`
4 Colour B	No		No	White	`-A1`	`RGB`
	No		Yes	Turquoise	`-A2`	`BG`
	Yes		No	Red	`-A3`	`G`
	Yes		Yes	White	`-A4`	`R`

Changing the colour decoding

Gweb DKU attributes

Attribute	value	Attribute	value
Red	R	White	RGB
Green	G	Yellow	RG
Blue	B	Violet	RB
Blink	K	Turquoise	BG
Inverse video	I	Underline	U
Low intensity	L	Hidden	H

To change one attribute to another:

Attribute on DKU	Desired Attribute	Parameter
White	Red	`-A1 R`
Turquoise	Violet, inverse video	`-A2 RBI`
Green	White, low	`-A3 RGBL`
Red	Green, underline	`-A4 GU`
Yellow	Underline	`-A5 U`
Blue	Inverse video	`-A6 I`
Violet	Low intensity	`-A7 L`
Black	White	`-A8 RGB`

Changing the field attributes

Parameters for changing attributes, DKU7211 colours, or fields:

`-An`	`rgbkiluh`	n from 1-8 DKU7211 colour attributes
`-AF`	`rgbkiluh`	Fixed fields
`-AV`	`rgbkiluh`	Variable fields
`-APRO`	`rgbkiluh`	Protected fields
`-AXMT`	`rgbkiluh`	Transmittable fields
`-APRI`	`rgbkiluh`	Printable fields
`-AFILL`	`rgbkiluh`	Must fill fields
`-AENT`	`rgbkiluh`	Must enter fields
`-ARJS`	`rgbkiluh`	Right-justified fields
`-APNUM`	`rgbkiluh`	Pure numeric fields
`-AQNUM`	`rgbkiluh`	Questar numeric fields
`-AVNUM`	`rgbkiluh`	VIP numeric fields
`-ASIGN`	`rgbkiluh`	Signed fields
`-ABDG`	`rgbkiluh`	Badge reader fields

To set an attribute for a field type e.g.

Field type	Desired attribute	Parameter
Fixed field	green	`-AF G`
Variable	violet, low	`-AV RBL`

3270 parameters

IBM 3270 emulation parameters should be set in the `–user` section of the `gweb.cfg` file: defaults in upper case.

Parameter		Description
`-An`	`rgbkilu`	Overrides base colour decoding, see ***Colours and attributes***.
`-DBG`	`on/OFF`	Turns on an emulation level trace.
`-II`	`on/OFF`	Ignore the 3270 Field Attribute: "intensified display". When set to ON, G3270 will not highlight characters.
`-LC`	`ON/off`	Lower case is transmitted to the host. Can be turned off so that all data to the host is transmitted in upper case (although it is displayed in lower case on the screen).
`-NUM`	`on/OFF`	Check numeric attribute. Most 3270 keyboards disable numeric checking.
`-QR`	`3278`	Response to host query. 3279 gives extended attributes and colour.
`-SX`	`on/OFF`	EBCDIC #@$ transliterate to [\] (Scandinavian transliteration).
`-XL`	`US`	Translation from *`Host Links`* (ISO/Do11) 8-bit characters to 7-bit equivalents to the host, and vice versa. The correct `-XL` (`GB`, `GE`, `FR`, `SF`, `DE`, `NO`, `SP`, `IT`, `JA`) must be specified if you choose an 8-bit profile in the `profiles` file and communicate with a 7-bit national host.
`-XX`	`hxhx`	Any incoming character from the host can be translated into any other for display purposes. Both are expressed in hex, and the first becomes the second. This is done after the EBCDIC => ASCII transliteration.

Colours and attributes

The 3270 terminal family has two modes of operation. In base colour mode the colours are generated locally using the attributes of the field on the screen. In extended colour mode the host application explicitly chooses the colours to be used.

Changing the base colour decoding

Gweb 3270 attributes

Attribute	Value	Attribute	Value
Blink	K	Turquoise	BG
Blue	B	Underline	U
Green	G	Violet	RB
Inverse video	I	White	RGB
Low intensity	L	Yellow	RG
Red	R		

The default field decoding for base colour mode:

Field Attribute	Colour	Parameter	Value
Unprotected, normal intensity	Green	`-A1`	GL
Unprotected, intensified	Red	`-A2`	R
Protected, normal intensity	Turquoise	`-A3`	GBL
Protected, intensified	White	`-A4`	RGB

To make unprotected, intensified fields violet and inverse video:

```
-A2   RBI
```

5250 parameters

IBM 5250 emulation parameters should be set in the `–user` section of the `gweb.cfg` file: defaults in uppercase.

Parameter		Description
`-DBG`	`on/OFF`	Turns on an emulation level trace.
`-EA`	`INT`	EBCDIC-ASCII conversion (default INT)
	`INT`	`International`
	`UK`	`English (UK)`
	`US`	`English (US)`
	`AG`	`Austrian/German`
	`AG1`	`Austrian/German (alt)`
	`BRA`	`Brazilian`
	`BE`	`Belgian`
	`CFR`	`Canadian French`
	`FR`	`French`
	`ICE`	`Icelandic`
	`SFI`	`Finnish/Swedish`
	`SF1`	`Finnish/Swedish (alt)`
	`DN`	`Danish/Norwegian`
	`DN1`	`Danish/Norwegian (alt)`
	`IT`	`Italian`
	`JAP`	`Japanese English`
	`PO`	`Portuguese`
	`SP`	`Spanish`
	`SPS`	`Spanish Speaking`
	`SP1`	`Spanish (alt)`
`-LC`	`ON/off`	Lower case is transmitted to the host. Can be turned off so that all data to the host is transmitted in upper case (although it is displayed in lower case on the screen).
`-NA`	`on/OFF`	Norwegian ASCII (default OFF)
`-XX`	`Hxhx`	Any incoming character from the host can be translated into any other for display purposes. Both are expressed in hex, and the first becomes the second. This is done after the EBCDIC => ASCII transliteration.

Line handler parameters (-LI)

All line handler parameters are described in the *G&R/Gline* manual.

Some DSA parameters

Parameter		Description
-LI	DSA:GATEWAY	Use DSA (or DIWS if historical reasons). Optionally connect via Ggate on GATEWAY.
-CO	CONAME	Use a CONAME in dsa.cfg rather than setting all line parameters here.
-PCO	CONAME	CONAME of a printer in dsa.cfg
-HM	DPS8/DPS7/ DPS6/CXI	DPS8 is the default. Use CXI for TP8.
-MN	mailbox name	For each active user, a unique mailbox name can be specified. Otherwise it is generated.
-DA	default application	Specifies the remote application, e.g. TP8, TSS, TDS or IOF.
-DX	default extension	Extension to mainframe application mailbox (CXI mode log on to TP8).
-DN	default node	Session control name of the mainframe.
-DU	default userid	Userid for connect letter to mainframe.
-D?	default password	Password for connect letter to mainframe.
-PW	default password	Password for connect letter to mainframe.
-DB	default billing	Billing for connect letter to mainframe.
-DP	default project	Project for connect letter to mainframe.
-UR	user record	GRTS ID/LID/user string.
-TM		Terminal type given to application.

The terminal type depends on the emulation being used. For DKU there are: DKU7007, DKU7107, DKU9107, DKU7211, VIP7700, VIP7760. For VIP7800 there are: VIP7801, VIP7802, VIP7804, VIP7814, TXT7801.

Some TNVIP parameters

Parameter		Description
`-LI`	`TCP`	Use TCP
`-AM`	`TNVIP`	Application mode
`-HO`	`hostname:port`	Numeric/symbolic IP-address, optional port
`-RES`	`mailboxname`	Resource name for TNVIP
`-TM`	`terminal_type`	Terminal type

For TNVIP the terminal type must be one of the synchronous types allowed by the protocol. For DKU these are: VIP7700, VIP7760, DKU7005, DKU7007D, DKU7105, DKU7107D, DKU7211, DKU7211D. For VIP7800 these are: VIP7804, VIP7804V, VIP7814, HDS7, VIP8800.

Some TN3270/TN3270E parameters

Parameter	Value	Explanation
`-LI`	`TCP`	Use TCP/IP
`-AM`	`TN3270`	or `TN3270E`
`-AP`	`on/OFF`	Select the printer LU name associated with the LU name used by the screen (`-LU`)
`-HO`	hostname:port	Numeric/symbolic IP-address, optional port
`-LU`	LUname	The TN3270 gateway uses this to map the connection to a specific LU.
`-TM`		Terminal mode
	`IBM-3278-2` `IBM-3278-3` `IBM-3278-4` `IBM-3278-5` `IBM-3279-2` `IBM-3279-3`	When connecting to IBM hosts over TCP/IP, the TN3270/TN3270E protocol negotiates one of these values for terminal mode. You must choose one that is acceptable to the IBM TCP/IP front end. All types may have -E appended to indicate extended attribute capability.

Example gweb.cfg with DSA hosts

```
# DSA connection via Ggate gateway on ggate.gar.no
# All DSA parameters are configured on the Ggate server
# in the section ‘coname tp8test’ of dsa.cfg.
# LIDs are assigned automatically from a pool
# Terminal Mode parameter can be picked up from HTML
# page (-unlock tm)
host tp8test
 -gweb
  -unlock tm
 -user
  -snd off
  -rw  10
 -li dsa:ggate.gar.no
  -co tp8test
..-pco tp8printer

# DSA connection directly from Gweb server to host
# en06, which has to be configured in local dsa.cfg
# configuration file.
# Terminal Mode parameter can be picked up from HTML
# page (-unlock tm)
host iofen06
 -gweb
  -unlock tm
 -user
   -snd off
   -rw  10
 -li dsa
  -hm dps7
  -du userid
  -pw passwd
  -dp project
  -db billing
  -da iof
  -dn en06
```

Example gweb.cfg with TN3270, TN5250

```
# TN3270 connection to standard telnet port (23)

host locis
-user
  -snd off
  -rw  20
 -li tcp
  -ho locis.loc.gov
  -am tn3270

# TN3270E connection to standard telnet port (23)
# TN3270E luname specified

host ibm
-user
  -snd off
  -rw  20
-li tcp
  -ho tn3270e.gar.no
  -am tn3270e
  -tm IBM-3278-2
  -lu TERM4

# TN5250 connection to standard telnet port (23)

host as400
-user
  -snd off
  -rw  20
-li tcp
  -ho as400.gar.no
  -am tn5250
  -tm IBM-3179-2
```

Example gweb.cfg with TNVIP and Telnet

```
# TNVIP connection to standard telnet port (23)
# No TNVIP resource name specified

host tnvip
 -gweb
 -user
   -snd off
 -li tcp
   -am tnvip
   -tm VIP7814
   -ho 129.182.2.13

# TNVIP connection to nonstandard telnet port (7323)
# TNVIP resource name specified

host iofen06
 -gweb
 -user
   -snd off
 -li tcp
   -am tnvip
   -tm DKU7105
   -ho tnvip.gar.no
   -rp 7323
   -res iofen06

# Telnet connection to standard telnet port (23)
# Can be used when connecting to HVX

host hvxtest
 -gweb
 -user
  -snd off
  -rw 20
 -li tcp
  -am telnet
  -hw 8
  -dn hvx.gar.no
  -tm glinkvip
```

The HTML startup page

You can check if your Gweb is installed correctly by asking ‘about Gweb’:

```
http://www.gar.no/cgi-bin/gweb?about
```

Choosing a session from the list

```
<h2>G&amp;amp;R mainframe session list</h2>
<a href="http://www.gar.no/cgi-bin/gweb">list</a>
```

A URL for Gweb with no parameters causes Gweb to return a list of the available sessions. Sessions are only included in the list if they are enabled using the `-emu` parameter in the host directive in the `gweb.cfg` file:

```
host <hostname> -emu <emulation> -title "title"
```

The `-emu` parameter specifies one of (`dku`, `7800`, `3270`, `5250`) and the optional `-title` is any descriptive text, otherwise `<hostname>` is used.

If the facility to track Gweb sessions by using HTTP cookies is enabled (default), the session list will also list all active sessions initiated by this browser. See the `"-sessioncookies"` configuration file parameter on page 35 for details.

Using a URL with parameters

```
<a href="http://www.gar.no/cgi-bin/gweb?mode=7800&sessionname=tp8v2">TP8 test</a>
```

The informational fields Gweb needs to start a session can be included in the URL. The absolute minimum is mode (`dku`, `7800`, `3270`, `5250`) and session name (`hostname`). See below for other fields.

Using an HTML form

A startup HTML page, in addition to informational text, must contain a form that will be submitted to Gweb in order to make the connection e.g.:

```
<h2>G&amp;amp;R VIP7800 session to TP8</h2>
<form method=POST action="/cgi-bin/gweb" target="_blank">
<input type=hidden name=sessionname value=tp8v2>
<input type=hidden name=mode value=7800></td>
<input type=submit name=cmd value="Connect">
</form>
```

Note that `target="_blank"` will open a separate window for Gweb.

Required fields

The URL or form **must** contain the following fields:

Field name	Contents	Description
`mode`	7800, 3270, 5250, or dku	Tells Gweb which emulation mode to use
`sessionname`	A host entry	Specifies a session name in the `gweb.cfg` configuration file

Optional fields

Field name	Contents	Description
`sysdir`	System Directory	Specifies the path to the G&R System Directory if the installation is not in the default location (UNIX/Linux only).
`hc`	keystrokes	Keystrokes in CPIC format that will be used for an initial transmission to the host. See also the `-hc` parameter.
`param`**`N`**	any text string	User-supplied parameter. **N** is an integer value from **1** to **50** e.g.: `<input type=hidden name=param5 value="VALUE FOR param5">` `<input type=text name=param13 value="">`

Gline parameter fields

Additionally, depending on the host, the form can contain several fields that allow the user to enter new values for line parameters. Line parameters that can be set from the HTML form (providing they've been unlocked in `gweb.cfg`) are:

Field name	Description
`CO, PCO`	Terminal (and printer) session `CONAME` in `dsa.cfg`
`DA`	Application mailbox for connect
`DX`	Application mailbox extension
`DB`	The billing account
`DN`	The remote host's node name
`DP`	The project name
`DU`	The user's login account name
`LU`	The 3270 Logical Unit (LU) name.
`MN`	The local mailbox name
`MX`	The local mailbox extension
`PW`	The user's password
`RES`	The TNVIP resource name.
`TI`	The terminal id
`TM`	The terminal mode
`UR`	The user record

Several sample pages are included in the Gweb delivery. The source files can be examined by viewing the source from your browser, or by accessing them via the file system. They are in the `gweb` sub-directory under the `html` directory in the Host Links system directory. The sample pages are available as a demonstration if you are accessing the G&R web server in Oslo to read this documentation.

The `3270.htm` and `5250.htm` sample documents present you with a logon page. These pages allow you to choose a session name from a drop-down list, and optionally set the line parameter that gives the address of a TN3270 (or TN5250) server. This sample page is live and you can test it. It will function if your web server is able to access the Internet. The demonstration is configured to make TN3270 and TN5250 connections directly to TN3270 and TN5250 servers.

The `7800.htm` and `dku.htm` sample documents allow you to pick an application from a drop-down list. These sample pages are live and you can test them. The demonstrations will work if your web server is able to connect over the Internet to *`G&R/Ggate`* running on the G&R web server in Norway. From there a DSA connection is made to the GCOS8 systems that host the demonstration systems. The HTML in the samples shows how *`G&R/Gline`* parameters can be entered dynamically from the form.

The `gwebline.htm` sample document allows you to set almost the whole range of *`G&R/Gline`* parameters in the connection form. The HTML in the sample shows how you can set very many of the *`G&R/Gline`* parameters. The choice of mode and terminal type are constrained by drop down lists, but all the others can be entered.

Facelifting

Gweb Professional Edition offers extended facelifting functionality for mainframe applications. Facelifting is done on a session basis, by recognizing specific application screens within the session, and initiating customized actions. If a screen is not recognized, the automatically generated HTML version of the screen is displayed. Applications can be facelifted screen by screen as convenient.

When Gweb recognizes a screen the customized actions can be: display of your own customized HTML page using data from the screen, automatic reply, skipping display of the screen on the browser or start of your own external script to take over the dialog. You can also change the configured attributes of the session to suit this particular dialog step.

Configuration

You configure the screen recognition criteria and the action parameters in a file `index.cfg`, residing in a Gweb directory specific to the session. For example, if the Gweb session is named `locis` the facelifting configuration file resides in the `locis` subdirectory in the `gweb` subdirectory under the `html` directory in the Host Links system directory.

```
/usr/gar/html/gweb/locis/index.cfg
```

The `index.cfg` file consists of two parts; one part defining patterns that identify individual host screens, and one part defining specific attributes and actions for each individual host screen identified.

The pattern entries define one or more texts in specific areas on a screen that, if present, allow Gweb to recognize the screen by the name given in the pattern entry. This name identifies the set of attributes that apply to the screen.

An attribute entry starts with the name of a screen recognized by a pattern followed by one or more attribute directives that change the presentation of this specific screen. A very important attribute directive `HtmlFile` points at a user written HTML file that will be displayed on the browser instead of the screen, but the attribute directives can also define automatic actions and a transmit back to the mainframe, skipping display of the screen entirely.

The `index.cfg` file also defines defaults for all host screens received, and optionally attributes for the dummy screen (`$*$dis`) that Gweb 'recognizes' when the session is disconnected.

Gweb includes the Gweb facelifting editor that helps you to identify host screens and build the `index.cfg` file.

Gweb includes a sample `index.cfg` file. This has patterns that recognize several screens sent from the `locis` host session example delivered with Gweb. The first screen is given a name `menu`, and a specific attribute entry later in the file points to the HTML file to be displayed on the browser when this screen is recognized. The remaining screens use their HTML facelifting file directly as a name, and do not need attribute entries pointing to the HTML. The default section specifies the HTML files that provide a header and footer for all screens that do not have a specific header and footer.

```
*
* Example Gweb facelifting configuration file
*

patterns:
        menu            LOCIS@68,24 MENU@73,24
        catalog.htm     CATALOG@71,24
        legislation.htm LEGISLATION1@68,24
        copyright.htm   COPYRIGHT@68,24
        braille.htm     BRAILLEAUDIO@68,24

default:
        PrefixFile     heading.htm
        PostfixFile    footing.htm
        Color #88cc00  white

menu:
        HtmlFile menu.htm

$*$dis:
        HtmlFile _disc.htm
```

Screen recognition

The description of screen recognition that follows is normally only needed in special cases. Normally screen recognition is automated (see ***Automating screen recognition***).

Simple pattern matching

The purpose of patterns is to give Gweb a way to uniquely identify a screen sent from the application. When the pattern(s) match, Gweb takes action as defined by the directives in the attribute entry for the screen name.

Pattern definitions look like this in the `index.cfg` file:

```
patterns:
   formname1   text@x,y [text@x,y ...]
   formname2   text@x,y [text@x,y ...]
```

The `text` in question is a string of characters on the screen, starting at the given coordinates. The `x` coordinate is the column, and `y` is the row. The top left screen coordinate is `1,1`. If more than one pattern is defined, the screen must match all the given texts. If the `text` contains a 'commercial-at' character ('@') it must be entered twice. Here is an example that allows Gweb to recognize a screen named menu because the screen has text `LOCIS` in column 64 of line 24 and text `MENU` in column 73 of line 24:

```
patterns:
   menu      LOCIS@68,24    MENU@73,24
```

Advanced pattern matching

The syntax of the coordinate specification at the right-hand side of the @ character is this:

```
@xMin[-xMax],yMin[-yMax][,options...]
```

where parameters in [brackets] are optional.

`xMin-xMax` means that the text in question can start somewhere in column `[xMin]` thru `[xMax]` on the line. Similarly, `yMin-yMax` means that the text is located on one of the rows `[yMin]` thru `[yMax]`. These can in fact be combined; `error@1-76,1-24` looks for the word "error" anywhere on the screen.

Pattern matching options

You can specify options to the pattern-matching algorithm. The options are specified as single characters following a comma after the row element of the coordinate. All options can be combined. The available options are:

Option I - Case independent comparisons

Normally, the most efficient way to match texts on a screen is to use exact match – that is, the match text you supply should match exactly the text on the screen. This option defines that the pattern matching should be performed in a case-insensitive manner.

Example: `error@1-76,1-24,i` looks for the word "error" or "ERROR" or "eRrOr" anywhere on the screen.

Option W - Wildcard pattern matching

In some circumstances it is not possible to look for a match on a fixed pattern. In this case, you may supply the W option that allows the pattern string to contain wildcard characters. The wildcard characters are defined in the table below.

Character	Description
\	Uses the literal meaning of the next character. Useful if you need to check for any of the characters below. Example: `Continue\?` matches the text `Continue` followed by a question mark. Without the \ character you would get a match on Continue followed by any character, not just a question mark.
?	Matches any single character. Example: `Galla?her` matches all of Gallagher, Gall her and GallaXher.
*	Matches any character zero or more times. Example: `G*b` matches all of Gweb, Gb and Gallagher & Rob.
`[...]` `[^...]`	Matches any character you supply inside the [brackets]. If the first character is a circumflex (^), this is negated and matches any character not in the [brackets]. You can supply a range of characters with a hyphen; e.g. `A-Z` matches all upper-case characters in that range. Example: Emulation: [37] matches both Emulation: 3270 and Emulation: 7800, but not Emulation: DKU.

Note that in Wildcard matching mode, all comparisons are done on the whole screen row. This means that your pattern is by default the whole row of 80 characters. You must start the pattern with an asterisk if it does not start in column 1, and you must end it with an asterisk if it does not end in column 80, e.g. if you are looking for the word Gweb somewhere within line 24:

```
*Gweb*@1,24,W.
```

Option C - Cursor position checking

In some situations, the only way you can distinguish one screen (or, rather, one **mode** of a screen) from another is by recognizing the screen, and additionally looking at where the host positioned the cursor. The Cursor Position checking option gives a match if the cursor position is within the given range. The `pattern` is ignored in this case. However, you still have to enter a pattern in order not to break the syntax rules. Example:

```
MYAPP@68,24 whatever@10,12-14,C
```

returns a match when the MYAPP screen is received, if additionally the cursor is positioned in column 10 of line 12, 13 or 14. The patterns are examined in the order they are defined in `index.cfg`, so the pattern for a screen and cursor position would be defined before the pattern for the same screen without the cursor position (if used).

Option V - Variable field checking

In some situations, the only way you can distinguish one screen (or, rather, one **mode** of a screen) from another is by checking if the host has created a variable field in a certain screen position. The Variable Field checking option gives a match if and only if the given range of screen coordinates encloses at least a part of a variable field. The `pattern` is ignored in this case. However, you still have to enter a pattern in order not to break the syntax rules. Example:

```
MYAPP@68,24 whatever@10,12-14,V
```

returns a match when the MYAPP screen is received, if additionally column 10 of line 12, 13 or 14 is within a variable field.

It should be noted that using *Advanced pattern matching* possibilities is more CPU demanding that *Simple pattern matching*.

Special-purpose settings

In addition to settings for screens that Gweb can identify with a pattern, some other "pseudo-screens" can be defined.

Default screen settings

Default: section settings apply to all received screens, provided the corresponding attribute is not defined for the specific screen. Example:

```
default:
  Fkeys Transmit, Disconnect, F1, F2, F3, F4, Bottom
```

Disconnection settings

$*$DIS: section settings apply when the session is disconnected. Gweb will by default display a built-in HTML page informing the user of the disconnection. You can override this by configuring your own HTML to be displayed here. Example:

```
$*$dis:
 HtmlFile logoff.htm
```

External script settings

Settings in `$*$1:` thru `$*$9:` sections are only used as an optional placeholder for resident external script configurations. Examples:

```
$*$1:
  Call O6 perl book.pl
$*$2:
  Call OI4 perl result.pl
$*$3:
  Call I7 perl validate.pl
```

These scripts can also be referenced from the `<input type=text name=InputScript value="">` mechanism that lets you specify directly in the HTML page the input script to be executed.

Attributes and actions

When a given pattern matches the content of the screen, Gweb will execute the directives in the attribute entry with the same name as the name of the screen defined in the pattern entry.

If no patterns match the screen, the directives in the (optional) "default" attribute entry will be executed.

If a pattern matches, but the screen name is missing from the configuration file, Gweb assumes that the screen name in the pattern entry is the name of an HTML file to be displayed. In other words the pattern below, with no attribute entry:

```
patterns:
      catalog.htm CATALOG@71,24
```

behaves as if there was an attribute entry as follows:

```
   catalog.htm:
      HtmlFile catalog.htm
```

The facelifting attribute directives are described below. Note that all attributes are optional. If not specified, the attributes from the `gweb.cfg` configuration file will be used. For example, if you do not specify an action that results in a transmit back to the application, and do not specify your own HTML file to be displayed on the browser, Gweb will generate an automatically face-lifted HTML page.

In all cases the attribute entries start with the name of a screen, followed by one or more directives. The screen names are as specified in the pattern directives, and additionally `default` and `$*$dis`.

Directive	Description
`HtmlBase directory`	Defines the directory where all HTML files are located, and is only used in the `default` entry. If not present, Gweb assumes that the directory containing the `index.cfg` file also contains the facelifting files.
`HtmlFile filename`	Defines the name of a file containing a face-lifted version of the host screen. When using the Gweb face-lifting editor to identify forms, the `filename` will point to the file that Gweb generated automatically.
`PrefixFile filename` `PostfixFile filename`	Defines names of HTML files that are used in front of or after the face-lifted HTML. If not supplied, Gweb provides standard headers and footers.
`PrefixCode HTML-code...` `PostfixCode HTML-code...`	Defines HTML code that is inserted in front of or after the face-lifted HTML.
`Cursor F.S`	Instructs Gweb to position the cursor in variable field number F, sub-field number S of the screen. The upper/left field of the form is "0.0". Sub-fields other than 0 occur only when a variable field extends over several lines (DKU only). If you use this feature you must either use the macro `#javascript(doFocus)#` or a combination of `#js(Field)#` and your custom `<body #js(Body)#` code in the header file (defined by the `PrefixFile` directive above).
`Field N set value` `Field N default value`	Instructs Gweb to pre-fill variable field number N with a certain value. If the `set` qualifier is used, the field content is set to the given value unconditionally. If the `default` qualifier is used, then the field will be pre-filled only if it was blank.

Directive	Description
`Color #rrggbb color-name`	Maps screen colors to HTML colors. The "color-name" is one of White, Turquoise, Green, Red, Yellow, Blue, Violet or Black, optionally combined with the text "High intensity" or "Low intensity". When the screen uses one of these colors, the RGB value of the desired color will be used when presenting data on the browser. The "#RRGGBB" notation describes the hexadecimal weight of the Red, Green and Blue color components in the range `00` to `FF`. White is normally defined as `#ffffff`, red is `#ff0000`, etc.
`Timeout N`	Specifies the number of minutes Gweb keeps an active session running before it is disconnected. If specified, it overrides the `-timeout N` directive in the `gweb.cfg` configuration file, which defines a general timeout for the session. Using this directive you can specify timeouts for specific host screens.
`Action action`	Specifies that a certain action should take place automatically. The `action` keyword is one of `F1` thru `F24`, `PA1` thru `PA3`, `Transmit`, `Break`, `Disconnect`, `SysReq`, `Test`, `RollUp`, `RollDown`, `FieldPlus`, `FieldMinus`, `FieldExit`, `Help`, `Return` and `Macro`. For `Macro` a CPI-C keycode sequence must also be defined using the `Macro` directive described below. Otherwise a simulated key press of the defined Action key will be sent to the application, and no HTML file will be displayed.
`Script cmd...` `Iscript cmd...` DEPRECATED (but still supported) see `Call`	The `Script` directive specifies that the screen data be passed thru your own script, before any output is sent to the browser. The supplied command is executed instead of Gweb's internal facelifting. The `IScript` command intercepts data from the browser before it is sent to the application.

Directive	Description
`Call SPEC command...`	This directive specifies that the screen data from the application, or input from the browser, be passed thru your own scripting routine. The script specification string (`SPEC`) consists of one or more of the characters `O`, `I` and `1-9`. The specification characters have the following meaning: **`O`** The script is an output script. After Gweb has received a screen, but before output is sent to the browser, the script is called instead of Gweb's internal facelifting. **`I`** The script is an input script. After Gweb has received data from the browser, but before anything is sent to the application, the script is called. Note that an input script also can be specified using the InputScript HTML form tag. **`1-9`** The script is a Resident script. It stays in memory throughout the session, and Gweb calls it for each dialog step. The opposite of a Resident script is a Oneshot script. These are loaded when needed, executed, and when they terminate Gweb analyzes the output. Regardless of the script type, the output from the scripts determines if Gweb sends commands back to the host, replaces Gweb's HTML output with the result from the script, or continues normal operations. Example: `Call O perl book.pl` executes the command perl book.pl when the specific host screen is received. Please check the chapter entitled External scripting on page 120 for details.

Directive	Description
`Fkeys keyname, keyname,... position`	Defines which function keys should be visible, and the location. The names of the function keys are in the table for `Fkey`, below. The position is defined by one of the keywords `Top, Left, Bottom` or `Right`. For example, `Fkeys Transmit, Disconnect, F1, F2, F3, F4, Bottom`
`Fkey N <CPIC-KEYCODES>` `Fkey keyname <CPIC-KEYCODES>` Keycodes are expressed in the HLLAPI-like syntax @x e.g. `Fkey 1 @0@T@T11976@E` `Fkey F1 @0@T@T11976@E`	Assigns a keystroke sequence to a function key, overriding the default. You can use the key number `N` or name e.g. `F13`. **1-12** `F1` thru `F12.` VIP and DKU emulations: Function keys F1 thru F12. IBM emulations: Function keys PF1 thru PF12 **13-24** `F13` thru `F24.` VIP and DKU emulations: Function keys SF1 thru SF12 (F1-F12, shifted). IBM emulations: Function keys PF13 thru PF24 **25-27** `PA1` thru `PA3.` IBM 3270: Function keys **32, 36, 37** The `Transmit`, `Disconnect` and `Break` buttons. `Break` is ATTN for IBM **38** `SysReq.` IBM emulations **39-45** IBM 5250 buttons: `TEST`, `RollUp` and `RollDown`, `FieldPlus`, `FieldMinus`, `FieldExit`, `Help`. **46** VIP emulation: `Return` You can also define function key sequences in `gweb.cfg`, but these apply to the whole session. The function keys defined here can be tailored to match a specific host form. The keystrokes are given to the emulation when you press the key. Check the discussion of CPI-C keyboard codes for details of the definition.

Directive	Description
`FkeyText n text` `FkeyText keyname text`	This directive assigns a text to a function key button. You can use the key number or name. See `Fkey` above for key numbers and names e.g. `FkeyText 1 Help` `FkeyText Transmit Commit changes` gives Function key 1 the text `Help` instead of the default value `F1`, and the Transmit key would have the button text `Commit changes`.
`Macro <CPIC-KEYCODES>`	This directive defines a CPI-C keycode sequence to be executed when the screen has been received from the host (but not yet sent to the HTML browser), and the 'Action' attribute is set to macro. If the macro terminates with a transmit (`@E`) then no HTML is displayed, and the screen is skipped.
`Disconnect <CPIC-KEYCODES>`	This directive defines a CPI-C keycode sequence to be executed when the host connection is about to be terminated. If present, it overrides any contents of the `-hd kkkkkk` disconnect macro in the `gweb.cfg` configuration file.
`Content-Type value`	**Specialists only**. This directive defines the content type of the document sent to the web browser. The default content type/subtype is "`text/html`".

Directive	Description
`-gweb PARAMETER [VALUE]` `-user PARAMETER [VALUE]` `-line PARAMETER [VALUE]`	You can dynamically change the current run-time options. Any parameter you supply here is sent to the appropriate subsystem; i.e. Gweb facelifting code processes `-gweb` parameters; the terminal emulation code processes `-user` parameters; the line module processes `-line` parameters. The parameters themselves are the same parameters as you can use in the `gweb.cfg` configuration file. Note, however, that not all parameters have any effect other than at initialization time; e.g. `-line tcp` is obviously not processed once the connection has been established. Example: `screen1:` `  -user -rw 20` `screen2:` `  -user -rw 90` This part of the `index.cfg` configuration file says that each time "screen1" is sent by the host, the Receive Wait parameter is set to 20 seconds; and each time "screen2" is sent by the host, the Receive Wait parameter is set to 90 seconds.

Automating screen recognition

There are two basic methods provided in Gweb for automating the process of recognizing the application screens: the JavaScript method and the Gweb facelifting editor.

The JavaScript method

To enable the JavaScript method, insert `-capture js` in the `gweb.cfg` configuration file.

Each application screen appears in an extra browser window. You identify the form by marking one or more areas of text with the mouse. The form is saved into the `index.cfg` facelifting database after you have supplied the logical form name.

The Gweb facelifting editor

The Gweb facelifting editor takes its input from captured screens, lets you mark the text strings that uniquely identify them and creates or updates the configuration file.

To enable the screen capture function, you must temporarily include the directive `-capture on` or `-capture1 on` in the `gweb.cfg` file and then run through the screens that are candidates for facelifting. Remove the directive afterwards.

The Gweb capture feature records both the plain text and the HTML version of the application screens for reference. You use Gwebedit on the captured screens to create the facelifting database (the `index.cfg` file) that defines the recognition criteria for each screen. The Gweb facelifting editor shows you each screen and lets you select interactively the strings that Gweb should recognize.

Using Gwebedit

Execute the command

```
gwebedit <screen-name>
```

where <screen-name> is the name of a captured text file; e.g.

```
gwebedit /usr/gar/html/gweb/locis/00000000.gcf
```

Use the cursor keys to move around in the captured screen. Identify the text that makes this screen unique compared to other screens, and mark the text using the Space key. You can mark more than one text string. You can clear all markings by pressing the Clear key (CTRL/HOME).

When finished, hit the TRANSMIT/GREY+ or simply the **W** key. At this point you are prompted to supply a unique name for the screen in question. The name must contain at least one letter (a-z), and cannot contain spaces or colons. Furthermore, some names are reserved (`default`, `$*$dis` and `patterns`, for example) as they have special meaning in the Gweb Facelifting Configuration file.

The captured files (`00000000.htm` and `00000000.gcf`) are renamed to the name you chose, and the facelifting configuration file will be updated (created if necessary) with a pattern entry for the name, and an attribute entry for the name, with an `HtmlFile` directive pointing to `name.htm`.

You can now manually edit the configuration file. If the target screen is not to be displayed at all, you replace the `HtmlFile` directive with an action directive. If the screen is to be customized for display you edit the Gweb-generated HTML file associated with the screen (`name.htm`).

Facelifting configuration file validation

You can use Gwebedit to validate your facelifting configuration file. If you execute the command:

```
gwebedit -validate index.cfg
```

where `index.cfg` is the name of the file you want to validate, Gwebedit prints any error messages on the standard error device and it's own perception of the configuration file on the standard output.

Facelifting a screen

You facelift a screen by building an index file with recognition criteria for the screen that assign it a name. You insert an attribute entry for the name with an `HtmlFile` directive pointing to a customized HTML file. This can all be built manually, or you can use Gwebedit to build the index and pointer, and generate the standard Gweb facelifting HTML. You can use this Gweb-generated HTML file as a start point or simply delete it and start from scratch.

The HTML that you define is a template where you can insert macros that pick up data from the application screen if used in your output, and that accept user input as data for the variable fields on the screen, to be returned to the application. There is a whole range of macros available for this and other tasks (see below).

Your customized HTML must allow the user to respond to Gweb, giving input from the facelifted screen that can be mapped to the input needed by the application, usually by supplying content to variable fields defined in the screen. There are two basic ways of doing this. Your HTML can define a number of hyperlinks, each of which refers to Gweb, but returns different parameters, or you can define a HTML form with fields for user data entry and buttons for actions. Both methods can be used in the same HTML page.

There are no restrictions on the HTML techniques you can use to obtain input from the user. The only restriction on your HTML is that it must assign values to the variable fields in the application screen, and give a command, default transmit, to Gweb.

A very simple example would be an application screen that defined a single variable field and needed a user input of 1,2 or 3 in the field to choose the application function.

Defining hyperlinks with parameters

Your HTML template is coded using Gweb macros e.g.:

```
<UL>
<LI><A HREF="#gweb#?f0=1&#get_data#" #js(Href)#>Copyright</A></LI>
<LI><A HREF="#gweb#?f0=2&#get_data#" #js(Href)#>Braille</A></LI>
<LI><A HREF="#gweb#?f0=3&#get_data#" #js(Href)#>Legislation</A></LI>
</UL>
```

`#gweb#`	inserts the virtual address of gweb
`f0=`	assigns a value to the first variable field in the screen
`#get_data#`	inserts Gweb session identification information
`#js(Href)#`	allows use of a hyperlink while in a Gweb session

The macros (`#gweb#` etc.) are explained in detail below. The default command to Gweb is transmit. A click on one of these hyperlinks returns the value 1,2 or 3 as the content of the single variable field, and causes Gweb to transmit the result to the application.

Defining an HTML form for user input

For the same simple case, your HTML template would be coded e.g.:

```
Please enter:<br>
1 for copyright<br>
2 for Braille<br>
3 for Legislation<br>
<form method=POST action="#gweb#" #js(Form)#>
<input type="text" name="f0" maxlength="1" size="1" value="" #js(field)#>
<input type=submit name=cmd value=Transmit #js(Button)#>
#post_data#
</form>
```

`#gweb#`	inserts the virtual address of gweb
`#js(Form)#`	calls JavaScript routines for form checking
`type="text"`	define a text entry field
`name="f0"`	content assigned to first variable field in the screen
`#js(field)#`	calls JavaScript routines for field checking
`type=submit`	define a button
`name=cmd`	sends a Gweb command
`value=Transmit`	command is transmit fields to application
`#js(Button)#`	calls JavaScript routines for button checking
`#post_data#`	inserts Gweb session identification information

The macros (`#gweb#` etc.) are explained in detail below. A click on the button ‘Transmit’ submits the form to Gweb with the user input in the first variable field, and an explicit command to transmit to the application.

Transmit using the ENTER key

In the currently supported browsers the ENTER key generally acts as a click on the first button defined in a form. It is therefore obviously good design practice to define the TRANSMIT button first, so that the ENTER key submits the HTML form. If you define BREAK or DISCONNECT first you will almost certainly cause unexpected problems for the majority of your users, who will expect the ENTER key to act as TRANSMIT.

The JavaScripts we deliver with Gweb enhance the ENTER key by causing it to work as TRANSMIT in situations where the browser might otherwise not TRANSMIT (submit the HTML form).

Fields recognized by Gweb

Cmd, Cmd:value

```
<input type=submit name=cmd value=transmit>

<input type=submit name="cmd:transmit"
value="Update customer number">
```

In the first format Gweb recognizes the value as being the command. In the second format Gweb takes the command from the name of the field, after the colon, and ignores the value, which can be any descriptive text. In both cases the value is used as the text on the button.

The commands accepted by Gweb are:

Command	Description
`f1-f24`	transmits the function key to the application (for IBM interpreted as PF1 - PF24). For terminals with shifted function keys add twelve for the shift e.g. Shift/F1 = F13.
`PA1-PA3`	transmits the IBM3270 PA keys.
`transmit`	transmits the data (variable fields) to the application.
`refresh`	updates the current screen with new application data.
`break`	transmits a break to the application (ATTN for IBM).
`disconnect`	disconnects Gweb from the application.
`SysReq`	transmits a system request for IBM emulations.
`Test, RollUp, RollDown, FieldPlus, FieldMinus, FieldExit, Help`	transmits the IBM5250 keys.
`Return`	transmits a carriage return for VIP emulations.

Note that the sequence normally transmitted for a key can be overridden, either by an `-fN macro` directive in the `gweb.cfg` configuration file, or by an `Fkey N macro` directive in the facelifting configuration file.

Macro:CPIC-keycodes

```
<input type=submit name="macro:@0@T@Thelp@E"
value="Help on this topic">
```

Updates the variable fields in the emulation, and delivers the CPI-C keycode sequence to the emulation. The example above uses CPI-C-keycodes:

```
@0 - Home
@T - Tab
@E - Xmit
```

It creates a button with text `Help on this topic` that places the word "help" in variable field number three, and then transmits to the host.

Init:CPIC-keycodes

```
<input type=hidden value="" name="init:@C">
```

SPECIALISTS ONLY: delivers a user-defined CPIC keycode sequence to the emulator, before the variable fields. Normally, this is not necessary, and the init:string is empty. The example clears the screen before updating the fields.

Cursor

```
<input type=hidden name=cursor value=3>
```

This field tells Gweb in which field the cursor should be positioned when transmitting the screen data. By default, the cursor is positioned in the last variable field of the screen. Some emulations report the cursor position to the application, and it is significant. The `macro:XXX` method described above will override this setting.

fN, fNsN

```
<input type="text" name="f0" value="" #js(field)#>
```

This is how you deliver the contents of HTML fields to the variable fields of the screen in Gweb. The names of the variable fields start with the letter `f` followed by the field number (starting with zero). Optionally you can add the letter s (subfield) and another number. The subfield, default zero, is only relevant when a variable field spans over more than one line of the screen, and the number indicates the subfield.

SimpleGwebUpload[.N]

```
<textarea cols=80 rows=5 name=SimpleGwebUpload>
Line one
Line two
Line three
</textarea>
```

You use the `SimpleGwebUpload` field to upload data to the application. The lines, separated by newline characters, are delivered to the emulation, followed by a transmit command that sends them to the application. The emulation must be in a state where it can accept the lines and the transmit command.

You can specify as many as 10 of these fields by appending `.N` (default 0):

```
SimpleGwebUpload.0.......SimpleGwebUpload.9
```

In case of multiple `SimpleGwebUpload.N` fields, they are transmitted in the sequence they are numbered.

InputScript

```
<input type=hidden name="InputScript" value=3>
<input type=hidden name="InputScript" value=myscript>
```

This field tells Gweb which Input Script it should execute when this screen is submitted. The field value is either the single-digit number (1 thru 9) of a Resident Script, or an arbitrary command line for the Oneshot input script you want to execute. See the discussion of the Facelifting Configuration file Call directive for details about Resident vs. Oneshot scripts. An Input Script specified in this manner overrides any Input Script settings in the facelifting configuration file.

Print

The print command is normally included as a parameter to a hyperlink that is only displayed if there is print pending. The default hyperlink used by Gweb is:

```
<P><A HREF="#gweb#?print&#get_data#" TARGET="_blank"
#js(Href)#>Process host print</A>
```

This instructs Gweb to deliver any outstanding print data in a separate print window using default encoding. The print is delivered as specified by the HTML in the file `sssss` defined in the `-prt sssss` directive in `gweb.cfg` (or the default, see `-prt`).

See the section entitled ***Print Support*** for details of the standard ways of handling 'text only' and 'transparent' print data.

The default encoding is Text mode. In this mode, all characters reserved for HTML syntax (" & < >) and high-ASCII will be quoted according to the '&entity;' HTML standard. Line feed and form feed formatting information from the application is preserved.

You can use other encodings. This is controlled by the print command parameter `penc=NN` where `NN` denotes the encoding you need. The available encodings are:

```
penc=0 = text (default)
penc=1 = qpr
penc=2 = binary
penc=3 = html

e.g.
<P><A HREF="#gweb#?print&penc=3&#get_data#"
TARGET="_blank" #js(Href)#>Process host print</A>
```

Encodings `text`, `qpr` and `html` are described in the macro ***#print ENCODING [NUMBER]#***".

Encoding 2, `binary`, causes Gweb to send the print data as a binary stream directly to the browser. You would process such data with an external program.

Fieldseparator

```
<input type=hidden name=Fieldseparator value=",">
```

SPECIALISTS ONLY: defines a specific CPIC keycode sequence to use as the separator between form fields. Normally, the field separator is the TAB character (hexadecimal 0x09, CPIC-code `@T`). In some circumstances, notably when you have constructed a HTML form to build a host command where the host has set the terminal in TEXT mode, you might need to replace the default separator character with your own.

Embedded macros in HTML

Facelifting an application works like this:

1. Gweb receives a host screen.

2. It looks in the facelifting configuration file to see if there is a pattern that identifies this screen and corresponding attribute entry.

3. If the screen is not recognized, Gweb automatically generates the standard face-lifted version.

4. If recognized, Gweb processes the attribute entry for the screen. If this points to a HTML file it processes that too, expanding the embedded macros, and then outputs the resulting HTML to the browser.

The embedded macro language is described below. Each macro must be enclosed within a pair of hash characters (#) in order to let the Gweb HTML parser recognize them. The # can be included as text if you double it - i.e. ## will display as one single #.

#field(FIELD NO.[,name])#

Inserts the data from the specified variable field into the HTML file. The first field is number zero. The `name` parameter is optional, check the chapter on stacked forms on page 117 for details. The field number parameter can be a counter register in the form `$N`.

#screen(ROW,COL,LENGTH[,name])#

Reads data off the screen and inserts it into the HTML file. The upper left corner of the screen is 1,1, and the lower right corner is 24,80. The `name` parameter is optional, check the chapter on stacked forms on page 117 for details. The Row, Column and Length parameters can be counter registers in the form `$N`.

#buffer(OFFSET,LENGTH[,name])#

Inserts up to LENGTH characters from offset OFFSET (starting from zero) of the screen buffer into the HTML file. The `name` parameter is optional, check the chapter on stacked forms on page 117 for details. The Offset and Length parameters can be counter registers in the form `$N`.

#vfield(ROW,COL[,name])#

Returns the number of the field (counting from zero) at the given screen coordinate. The upper left corner of the screen is 1,1, and the lower right corner is 24,80. If there is no variable field in the given position, the macro returns a null string. The `name` parameter is optional, check the chapter on stacked forms on page 117 for details. The Row and Column parameters can be counter registers in the form `$N`.

#param(NUMBER)#

Inserts the contents of a user-supplied parameter into the HTML file. If the parameter is not defined, or the parameter number is invalid or out of bounds, the macro returns a null string. See the discussion on user-defined parameters on page 128. The Number parameter can be a counter register in the form `$N`.

#cursor(ROW,COL)#

Returns information about the cursor position. Depending on the ROW and COL parameters (which can be counter registers `$N`), there are four different types of return:

ROW	COL	Description
0	0	If the host put the cursor inside a variable field, return the field number (starting with zero)
0	> 0	Return the Y coordinate of the host's cursor position
> 0	0	Return the X coordinate of the host's cursor position
> 0	> 0	Return a non-null value if the supplied coordinate equals the host's cursor position

#getenv(ENV_VAR)#

Inserts the contents of the environment variable `ENV_VAR` into the HTML file. If the environment variable is not defined, nothing is inserted. A list of available environment variables can be found at

`http://hoohoo.ncsa.uiuc.edu/cgi/env.html`

Example: In a Gweb session on G&R's web server, the macro `#getenv(HTTP_HOST)#` would result in the string `www.gar.no`.

#format SPEC field#

Applies a format specification to the field in question. The field is either `screen(ROW,COL,LENGTH)` or `field(FIELD NO.)` as described above. The format specification is one or more characters that indicate a format to be applied. The supported format specification characters are:

`l`	Converts the field to lower case characters
`u`	Converts the field to UPPER case characters.
`n`	Converts the field to Name format, where the first character of each space-separated word is in Upper case and the rest is in lower case.
`t`	Trims (removes) leading spaces from the field.
`c`	Compacts the field by replacing any multiple occurrences of space characters by one single space. This implies Trim as described above.
`.`	Replaces one or more spaces by period (`'.'`). This implies Compact as described above.

Example:

If the contents of the screen in position 1,1 is `BILL  GATES` then the macro:

```
#format n. screen(1,1,40)#
```

would result in the string `"Bill.Gates"`.

#scratr field#

Returns a character string representing the visual attributes for the first character position of field. The field is either screen(ROW,COL,LENGTH) or field(FIELD NO.) as described above. The returned string consists of zero or more characters from the following set:

R	The color includes a RED component.
G	The color includes a GREEN component.
B	The color includes a BLUE component.
U	The Underline attribute is set.
F	The Flashing (blinking) attribute is set.
L	The Low intensity (Faint) attribute is set.
I	The Inverse attribute is set.
H	The Hidden (inviusible) attribute is set.
V	This is a Variable field.

Example: if the visual attribute of position 1,1 of the screen is Inverse, Underlined and Yellow (Red+Blue), then the macro command

```
#scratr screen(1,1,1)#
```

would result in the string

RBUI.

#html color [screen(r,c,l)]#

Maps the color and other visual attributes (e.g. blinking, inverse, underline) from the given part of the host screen (identified by the last parameter) to the HTML equivalents. If the last parameter is omitted from this macro call, the HTML attributes are reset.

Example: if the visual attributes of position 1,1 of the screen are Underlined and Green, then the macro command:

```
#html color screen(1,1,1)#TEXT HERE#html color#
```

results in the HTML string:

```
<FONT COLOR="#008000"><U>TEXT HERE</U></FONT>.
```

#html buttons#

Inserts HTML code for all buttons a standard screen should contain.

#html button N [buttontext]#

Inserts the HTML code that creates a button with all the necessary attributes and parameters. This macro is really just shorthand for

```
<input type=submit class="..." name="cmd:..."
value="BUTTONTEXT" #js(Button)#>
```

where all the HTML field attributes are supplied by Gweb, e.g.:

```
#html button 1 "Call help"
```

in a customized HTML template might expand to:

```
<input type="submit" class="Button1" name="cmd:F1"
value="Call help" #js(Button)#>
```

#get_data#, #post_data#

Inserts Gweb's session identification fields into the HTML. You always need one of these, get_data for URL's and post_data for forms.

```
<A HREF="#gweb#?f0=1&#get_data#" #js(Href)#>Copyright</A>
```

This inserts Gweb session identification into the URL.

```
Please enter:<br>
1 for copyright<br>
2 for Braille<br>
3 for Legislation<br>
<form method=POST action="#gweb#" #js(Form)#>
<input type="text" name="f0" maxlength="1" size="1" value="" #js(field)#>
<input type=submit name=cmd value=Transmit #js(Button)#>
#post_data#
</form>
```

This inserts Gweb session identification into an HTML form.

#include <mode> name [...]#

```
#include file filename [...]#
#include program program-name [...]#
```

This inserts the contents of a file, or the STDOUT results of an external program, into the Gweb HTML output. If the file or program does not exist, no action is taken. Parameters can be literals or any other facelifting macro that returns a value.

For mode `file` the name of the file is composed of the concatenation of all the parameters. Parameters with spaces must be quoted; otherwise each space is regarded as a parameter separator. Some examples:

`#include file /disclaimer.inc#`

Includes the file `/disclaimer.inc`.

`#include file c:\temp\ gweb(sessionname) .htm#`

Includes the file c:\temp\sessionname.htm, where `sessionname` is the session name returned by `#gweb(sessionname)#` as described later in this chapter. Note the concatenation without spaces.

`#include file field(1)#`

Includes a file that has the same name as the contents of the second variable field on the host form.

For mode `program` the first word is the name of the program, followed by a space, and then concatenation of parameters. Parameters with spaces must be quoted. Spaces between parameters must be quoted.

`#include program uname -a#`

Includes the STDOUT result of running program `uname -a`.

`#include program uname -a " " -b#`

Includes the STDOUT result of running program `uname -a -b` without the quoted space it would have been `uname -a-b`

The current directory when Gweb runs is:

`/usr/gar/html/gweb/[sessionname]`

If that directory does not exist, the current directory is:

`/usr/gar/html/gweb`

#execute name [...]#

Executes an external program. The program\s STDOUT output, if any, is discarded. See macro `#include program name#` for details.

#date [format]#

Inserts the current date and time. If a format specifier is not supplied, the date/time is inserted as `"YYYY-MM-DD HH:MM:SS"`. You can include your own text in the format string; the default would be coded:

```
Date: #date "%Y-%M-%D %H:%m:%s"#
```

Format	Description
%Y	Year, 4 digits
%y	Year, 2 digits
%M	Month, 2 digits
%D	Day, 2 digits
%d	Day, 1 or 2 digits
%H	Hour, 24-hour clock, 2 digits
%h	Hour, 12-hour clock, 2 digits
%P	AM or PM
%m	Minutes, 2 digits
%s	Seconds, 2 digits
%A	Month Name, full
%a	Month Name, abbreviated
%N	Day Name, full
%n	Day Name, abbreviated
%J	Julian date, 5 digits (YYDDD)
%j	Julian day, 3 digits (DDD)
%Z	Time Zone Name
%U	UTC time extension { '+-hhmm' }
%%	The character '%' itself

#gweb(VARIABLE)#

Inserts the contents of an internal Gweb variable. The variables are:

Variable	Description
`config parameter name`	Inserts the value of a configuration parameter, for example `-rs` or `-dl`. Not all are available: e.g. `#ifnnull gweb(-dl)#` `<META HTTP-EQUIV="Refresh"` `CONTENT="#gweb(-rs)#;URL=#gweb(-dl)#">` `You will automatically return to the` `<a href="#gweb(-dl)#">Gweb home page</a>` `in #gweb(-rs)# seconds.` `#endif#`
`formname`	Inserts the facelifted Gweb form name. If not facelifted (other than the automatic Gweb facelifting), a null string is returned. Otherwise the name of the form as identified in the `index.cfg` facelifting configuration file is returned. e.g. to check the facelifting status of the current form: `#ifnull gweb(formname)#` `  <!-- Gweb's standard facelifting -->` `#else#` `  #if "default" == gweb(formname)#` `   <!-- the 'default' facelifting -->` `  #else#` `   <!-- a manually facelifted page -->` `  #endif#` `#endif#`
`formnumber`	Inserts the number of the current form. The first form number processed by Gweb is zero, and it is incremented by 1 for each form received.
`browser[,len]`	Inserts the make of the browser in use. An optional parameter appends the major (len=0) and minor (len>0) browser version number.
`script`	Inserts the virtual location of the `gweb` program, as in the `SCRIPT_NAME` environment variable for CGI programs. Default `/cgi-bin/gweb`, but you can use `#gweb(script)#` (or short form `#gweb#`).
`sessionname`	Inserts the Gweb session name.
`sid`	Inserts the session identifier.
`title`	Inserts the Gweb window title.

#javascript(P)#, #js(P)#

Inserts information needed by the JavaScript delivered with Gweb to activate the keyboard, and calls to the JavaScript routines that you must use in the appropriate places in your customized HTML if you want the same functionality as Gweb provides for automatically facelifted screens. The different parameters are described in the table below.

Parameter	Description
`#js(variables)#`	Defines a set of JavaScript variables used by the externally defined `gweb.js` JavaScript program supplied with Gweb. By default this macro is defined in the gwebhead.htm file: `<SCRIPT LANGUAGE="JavaScript"> <!--` `#javascript(variables)#` `// --> </SCRIPT>` `<SCRIPT LANGUAGE="JavaScript"` `SRC="/gweb/gweb.js"></SCRIPT>` The variables define a number of aspects of the application screen, such as cursor position and a table of variable fields with names, widths and attributes. The external `gweb.js` JavaScript uses these variables when performing functions such as field validation, attribute checking, automatic tabulation, Enter=Transmit and so on. The JavaScript routines are called from tags inserted by the other `#js(parameter)#` macros, which you should use in order to engage the JavaScript routines for your customized HTML.
`#js(Body)#` `#js(Form)#` `#js(Href)#` `#js(Field)#` `#js(Button)#`	Use these to insert the calls to the JavaScript routines. You can override the standard macros using Gweb configuration directives `-jsbody`, `-jsform`, `-jshref`, `-jsfield` and `-jsbutton`, respectively.

#cookie(COOKIENAME)#

Inserts the contents of a cookie with the given name.

#cpic ...#

When this macro is executed, the CPIC keystrokes given as parameters to this macro are fed into the emulation engine and sent to the host. No output is sent to the browser. Example:

```
#cpic @0 @F end " " session @E #
```

All parameters are concatenated, and spaces removed. To enter a space in the string enclose it in single or double quotes.

#setparam N ...#

When this macro is executed, the text given as a parameter to this macro is assigned as the current value of the user-defined parameter given by **N** (**N** is a non-negative number in the range 1 thru 50). Example:

```
#setparam 14 "This is the value of parameter 14"#
```

All parameters are concatenated, and spaces removed. To enter a space in the string enclose it in single or double quotes.

#sessions(count|N)#

This macro is only meaningful if the system that records session cookies is enabled; see the "-sessioncookies" parameter on page 35.

The first format, `#sessions(count)#`, returns the number of additional sessions available. For example, if the current session is the only one, `#sessions(count)#` returns 0.

The second format, `#sessions(`**N**`)#`, (**N** is a non-negative number) returns the necessary HTML code to produce a hyperlink to the session in question. For **N**=0 the hyperlink is for the current session; for **N**=1 the first additional session, and so on.

#print(QUERY)#

Returns a decimal number as result of a query to the in-line print subsystem. The queries are:

Query	Description
`pending`	Returns the number of pending print jobs. If no print outputs are available, returns a null string.
`spooling`	Returns an 'S' if print data is arriving but is not yet finished. If no print is arriving, a null string is returned.
`first`	Returns the number of the first unprocessed print output.
`last`	Returns the total number of print outputs available.

A typical use of these macros might be to test if there is print waiting to be printed, and if so include a URL that will (if clicked) display print data in another window:

```
#ifnnull print(pending)#
<P><A HREF="#gweb#?print&#get_data#" TARGET="_blank"
#js(Href)#>Process host print</A>
#endif#
```

In this case, the print data will be processed using the `-prt printfile` control. See also the chapter discussing print on page 129.

#print ENCODING [NUMBER]#

Includes print data belonging to application print job numbered NUMBER encoded using encoding scheme ENCODING. The different encoding schemes are:

Encoding	Description
`text`	For 'text only' print. The print data, after emulation, is encoded as standard HTML entities for the browser, but preserves the line feed and form feed formatting information. By default the emulations transliterate print from the application character set to ISO8859-1, but can also handle other character sets. The browser character set must in these cases be chosen to match (`-httpcset` in `gweb.cfg`).
`html`	The print data is sent exactly as is to the browser. This setting must only be used if the application produces print output in HTML format.
`qpr`	For 'transparent' print containing binary print control sequences. The print data is encoded using the `Quoted-Printable` encoding scheme, as described in RFC2045 chapter 6.7, except that soft line breaks are not used.

The optional NUMBER parameter defines which of the available print jobs that should be printed. By default, all pending print will be included:

Number	Description
`pending`	Same as the default value: Print all pending (unprinted) data.
`current`	Print the first non-printed host print job.
`all`	Print all host print data jobs, even those that have already been printed.
`N`	Print host print job number **`N`**.
`N,M`	Print host print jobs from number **`N`** thru number **`M`**.

See also the chapter discussing host print on page 129.

#$N [OPERATOR value]#

Accesses the counter registers. There are 10 counter registers, named **$0** thru **$9**. Each holds a signed value between -2.147.483.648 and +2.147.483.647.

The macro form **#$N#** displays the content of counter register number **N**.

The macro form **#$N OPERATOR value#** performs an arithmetic function on counter register number `N`. In this case, nothing is displayed.

The functions are:

Macro	Description
`$N = value`	Assignment
`$N + value`	Addition
`$N – value`	Subtraction
`$N * value`	Multiplication
`$N / value`	Division
`$N % value`	Modulus
`$N ^ value`	Exponent
`$N & value`	Logical-AND
`$N | value`	Logical-OR

These counter registers can be used in other Gweb macros that require numeric parameters, such as `#param()#`, `#screen()#` and `#field()#`. They can also be used in program logic macros `#if#`, `#switch#` and `#loop#`.

#if#, #ifnull#, #ifnnull#

Lets you test the value of screen or variable data and change the contents of the HTML file accordingly. The #if ...# statement has the following syntax:

```
#if operator1 comparision operator2#
[ #elsif# ]
[ #else# ]
#endif#
```

where operator1 is typically a text, operator2 is typically another text or the contents of another Gweb macro, and comparision is one of the following:

Comparision	Description
`== or =`	operand1 must match operand2
`!= or <>`	operand1 must not match operand2
`in`	operand1 must occur within operand2 (case insensitive match)
`IN`	operand1 must occur within operand2 (case sensitive match)
`!in`	operand1 must not occur within operand2 (case insensitive match)
`!IN`	operand1 must not occur within operand2 (case sensitive match)

For our TP8 demo application, error codes sometimes appear in the lower left corner. You can test for their presence and insert them into your HTML code with:

```
#ifnnull screen(24,0,40)#
Error! #screen(24,0,40)#
#endif#
```

Another example that displays a Norwegian flag if field number five equals NO:

```
#if "NO" = field(4)#
<img src="/images/flags/#format l field(4)#.gif">
#endif#
```

The ELSIF statement is optional. If present, it has a similar syntax to the `#if ...#` statement, and is evaluated if the preceding IF test (and all preceding ELSIF tests, if any) were FALSE.

The ELSE statement is optional. If present, the text following the ELSE macro is evaluated if the IF test was FALSE.

The full syntax of the IF statement is as follows:

```
   #if operator1 comparision operator2#    ...
 [ #elsif operator1 comparision operator2# ... ]
 [ #elsif operator1 comparision operator2# ... ]
 [ #else#  ... ]
   #endif#
```

IF statements can be nested to a depth of 30.

#switch #

Lets you compare the value of screen or variable data with a predefined set of values, similar to the C-language construct with the same name, except that only the first matching case is evaluated.

The syntax of the SWITCH statement is as follows:

```
   #switch operator1#
   #case operator2# ...
 [ #case operator3# ... ]
 [ #case default#  ... ]
   #endswitch#
```

operator*N* is typically a text or the contents of another Gweb macro. For each `#case ...#` statement, *operator1* is compared with the value specified in the opening `#switch ...#` statement. If the two values match exactly, the text following the statement will be evaluated, until the next `#case ...#` or until the `#endswitch#` statement, which terminates the whole construct.

The `#case default#` statement is a catch-all with text to be evaluated if none of the preceding `#case ...#` has been evaluated.

SWITCH statements can be nested to a depth of 30.

#loop $N,v1,v2[,v3]#

Defines an iterative loop.

Before the first iteration of the loop, **`$N`** is initialized with the value `v1`.

Statements between the `#loop...#` and `#endloop#` are executed repeatedly as long as the counter **`$N`** does not exceed the value of `v2`.

By default, the **`$N`** counter is increased by 1 after each loop. You can override this value by using the `v3` parameter. Note that `v3` may be a negative value.

Example:

```
#loop $3,10,15#
   #ifnnull screen($3,1,10)#
      <P>Item name: #screen($3,1,10)#</P>
   #endif#
#endloop#
```

Facelifting macros example

A simple HTML file for facelifting our TP8 test application looks like this:

```
<h1>Gallagher & Robertson</h1>
<h2>Opening up the world from your desktop!</h2>
<h3>Welcome to GAR-WS on System-J in Phoenix</h3>
<table>
<tr><td>GCOS8 release:
<td>#screen(8,34,7)#
<tr><td>TP8 release:
<td>#screen(9,34,6)#
</table>
<p>
<table width=50%>
<tr><td>
<font size=-1>This TP8 system is for demonstration only. Its
purpose is to demonstrate GCOS connectivity from the Internet
using various G&R products.</font>
</table>
<p><a href="#gweb(script)#?cmd=Transmit&#get_data#"
#js(Href)#>Click</a> to continue.
```

You can see most of the text is static, but variable screen data is put into the file using the `#screen(ROW,COL,LENGTH)#` macro. At the bottom there's a hyperlink that instructs Gweb to simply transmit (there is a variable field in the screen, but it isn't used in the HTML). The `#get_data#` macro adds Gweb session identification data to the hyperlink.

If you wanted to display the input field in the demonstration screen, you could make an HTML input field to match it with the HTML tag:

```
<input name=f0 value="#field(0)#">
```

The macro `#field(0)#` instructs Gweb to place the variable data the host sends in field 0, into this field on the web page. When the form is sent back to Gweb, it looks for fields named `f0`, `f1`, `f2`, etc., and places whatever the user typed into those fields into the corresponding fields on the host screen.

Working knowledge of HTML is, of course, required.

The screen stack

The Gweb screen stack is an area where previous N application screens are kept. All details about the screen are saved, including the screen contents, width, height, status line and information about variable fields and their contents.

When Gweb receives a new screen it saves the previous screen on top of the stack. The oldest screen is lost. You can adjust the number of screens that are kept in memory with the `-stack N` parameter. There is a memory penalty of at least 4K (depending on the screen's complexity).

Usage

Stacked screens can be accessed by ordinary facelifting macros (see page 91) as discussed below:

Macro	Description
`#field(N,name)#`	If the name parameter is supplied, it indicates the name or number (see below) of the stacked screen where the field contents are to be retrieved. If the indicated screen does not exist in the stack, the macro returns a Null string.
`#screen(Y,X,L,name)#`	If the name parameter is supplied, it indicates the name or number (see below) of the stacked screen where screen content is to be retrieved. If the indicated screen does not exist in the stack, the macro returns a Null string.
`#buffer(OFF,L,name)#`	If the name parameter is supplied, it indicates the name or number (see below) of the stacked screen where screen content is to be retrieved. If the indicated form does not exist in the stack, the macro returns a Null string.
`#vfield(Y,X,name)#`	If the name parameter is supplied, it indicates the name or number (see below) of the stacked screen from which the variable field is to be retrieved. If the indicated screen does not exist in the stack, the macro returns a Null string.

You refer to a screen in the stack in one of two methods:

Referral by number: The screen selected in the stack is the N'th last received. For example, to pick up field number 5 from the 3rd last screen received from the application, use the macro `#field(5,3)#`.

Referral by name: You can select any screen in the stack by name if it has a name in `index.cfg`. For example, if your facelifting file contained the following definition:

```
patterns:
    menu       LOCIS@68,24     MENU@73,24
```

then you could refer to this screen with the macro

```
#screen(24,10,5,menu)#
```

Development macros

The `#dump#` macros can be useful while developing facelifting projects.

#dump fields [filename [...]]#

Dumps information about the variable fields to the HTML output or on a disk file `filename`. If the file does not already exist it is created; otherwise new data is appended to it. The filename can be specified as a string of parameters that are concatenated to give the name. Refer to the `#include file filename#` macro for details. The first line of the file will contain a decimal number indicating the number of variable fields. The next lines contain the actual variable field data, one per line, terminated by a newline e.g.:

```
4

text in first field

text in third field
text in fourth field
```

There was no text in the second field.

#dump <mode> [name]#

Dumps information indicated by `mode` from the current screen or a screen in the stack into the HTML output. The name, if used, picks a screen in the stack, and is the name from `index.cfg` or `1 - N`:

SCREEN a 24 or 72 line x 80 character image of the current screen

HEX as for screen, but in hex

ATTRIBS hex dump of the attribute bytes for the screen image

For example, to dump the last four forms, include the following in your gwebfoot.htm footer file:

```
#dump screen#
#dump screen 1#
#dump screen 2#
#dump screen 3#
```

#dump stack#

Dumps information about the stack into the HTML output e.g.:

```
Stack depth: 9
   1: name='(no name)'
   2: name='(no name)'
   3: name='(no name)'
   4: name='(no name)'
   5: name='(no name)'
   6: name='(no name)'
```

The `no name` above is replaced by the screen name from `index.cfg` if the screens are facelifted.

External scripting

Gweb can be configured to pass the screen data from the application and/or browser data thru your own scripting routines.

In the first case, immediately after Gweb has received data from the application, but before any output is sent to the browser, your own command is executed instead of Gweb's internal facelifting.

In the second case, your own command is executed when Gweb receives data from the browser that would normally be analyzed by Gweb and passed on to the application in the form of keystrokes.

Depending on the resulting output from the script, Gweb can send commands back to the application, replace Gweb's HTML output with the result from the script, or continue normal operations.

Writing an external script

You can either use the Perl scripting module supplied with Gweb, or you can use the scripting language of your choice. In the latter case, you will have to analyze and decode the Gweb scripting input file yourself, to extract the information needed. In any case you need to send the result of your script back to Gweb using the defined interface.

The external script interface: input data

The command you specified in the `index.cfg` facelifting configuration file is executed as is, except that for One-shot scripts a parameter is appended to the end of the command. This parameter is the name of a file that contains data that your script routine needs. This file contains lines of text in the following format:

V:variablename=value

This defines a variable and assigns it a value. A number of these will occur in the script input file, the first one will always be the Gweb release number, for example `V:Version=6.3.0`. See the table below for a detailed description of the possible variables and their meaning.

N:count (Output scripts only)

This line defines the number of variable fields on the screen. If there are no variable fields, the line will read `N:0`.

n:line,col,width,attribut (Output scripts only)

This line contains details about variable field number *n*. The first field is numbered 0 (zero), and the last field's number is N-1 (where N is defined in the `N:n` line as described above). For each variable field you get the following information (all decimal values):

Field	Description
`line`	The line number where the field starts (starting with zero)
`column`	The column number where the field starts (starting with zero)
`width`	The width of the field
`attribute`	The attribute of the first position of the field (hidden, inverse, and so on).

S:height,width (Output scripts only)

This line marks the beginning of the screen data, and defines how many lines of screen data follow, and the width of the lines.

The screen data follows immediately after the `S:height,width` line. The screen data is exactly as it would appear on a terminal screen.

P:post length (Input scripts only)

This line marks the beginning of the CGI variables posted by the browser, and defines how many bytes follow. The CGI data follows immediately after this line. The input is exactly as the browser transmitted the data to Gweb. Note that all parameters are decoded automatically by Gweb and supplied in the format `V:variablename=value`, so you probably do not need to analyze this data yourself.

File Separator (Resident scripts only)

The File Separator is a single line containing the FS character (hexadecimal 0x1C) followed by a newline. The purpose is to mark the end of input data from Gweb to the resident script. The resident script must also use it to indicate the end of the current output data.

Description of input variables

Variable name	Description
`Version`	Contains the Gweb version number.
`ScriptType`	Contains a decimal number that identifies the type of script invoked: **Value 0:** An "output" script, configured with the `Call SPEC command`.. statement in the index.cfg configuration file, that is invoked after Gweb has received data from the aplication but before HTML code is sent to the browser. **Value 1:** An "input" script, configured with the `Call SPEC command`.. statement in the index.cfg configuration file, that is invoked after Gweb has received data from the browser but before CPIC keystrokes are sent to the application.
`ScriptNumber`	Contains a decimal number that identifies the script residency. If present, this number indicates that this is a Resident script (not a One-shot script), and it is the same number assigned by you to the script using the `Call SPEC command`.. directive in the `index.cfg` facelifting configuration file.
`TempFile`	Contains the pathname of a temporary file that your script can use as needed. Gweb guarantees that it will not be deleted between host transactions. It will, however, automatically be deleted when your current session ends (i.e. when you disconnect). Typical usage of this file is to save persistent data that your scripts need to keep between transactions.
`TempDir`	Contains the pathname of a directory you can use for scratch data.

Variable name	Description
`CallCounter`	Contains a number that increases with 1 each time an external script is executed. The initial value is zero.
`FormId`	Contains a number that increases with 1 each time Gweb receives a screen. The initial value is zero.
`ScriptName`	Contains the name of the currently executing Gweb program. This is useful when your scripts produce HTML code with hyperlinks to the current host session; you would typically have code such as: `<A HREF="ScriptName?f0=VALUE&GetData">`. See also the `GetData` and `PostData` variables below.
`SessionName`	Contains the current session name.
`SessionId`	Contains the current session ID.
`Emulation`	Contains the emulation in use (3270, 5250, 7800, dku).
`GetData` `PostData`	Contains Gweb's session-specific data – you always need one of these (which one you use depends on how the form is built up).
`HostForm`	Contains the name of the current screen – the same name as in the `index.cfg` facelifting configuration file.
`SessionTitle`	Contains the session title.
`HostCursor`	Contains the x,y position of the cursor.
`TerminalMode`	0: Forms mode, 1: Text mode 2: Character mode
`ParamN`	User parameter N, one for each defined parameter
xxx	In the case of input scripts, all CGI parameters for Gweb are decoded, and are available here.

Example input file

```
V:Version=6.3.0
V:TempFile=c:\gar\config\system\temp\2032.gww
V:TempDir=c:\gar\config\system\temp\
V:ScriptType=0
V:CallCounter=0
V:FormId=2
V:ScriptName=/cgi-bin/gweb
V:SessionName=tp8v2
V:SessionId=2032
V:Emulation=7800
V:SessionTitle=tp8v2 - Gweb
V:GetData=sid=2032.BDODAGOB.3
V:PostData=<input type="hidden" name="sid" value="2032.BDODAGOB.3"><input type="hidden"
name="keyboard">
V:jsButton=onFocus="doSelect(this);"
V:jsForm=onSubmit="return doForm(this);"
V:jsHref=onClick="doClick(this);"
V:jsBody=onLoad="doFocus()" onKeyDown="return doKeyDown(event);" onUnload="doCleanup()"
onBeforeUnload="doUnload()"
V:jsField=onKeyPress="return doKeyPress(event);" onKeyUp="return doKeyUp(event);"
onFocus="doSelect(this);"
V:HostCursor=37,12
V:TerminalMode=0
V:REMOTE_ADDR=193.216.112.200
V:REMOTE_PORT=3296
V:REQUEST_METHOD=POST
V:SERVER_PROTOCOL=HTTP/1.1
V:SCRIPT_NAME=/cgi-bin/gweb
V:GATEWAY_INTERFACE=CGI/1.1
V:SERVER_SOFTWARE=G&R Gweb/6.3.0
V:SERVER_NAME=mats2
V:SERVER_PORT=80
V:COMPUTERNAME=MATS2
V:OS=Windows_NT
V:SystemDrive=C:
V:SystemRoot=C:\WINDOWS
V:windir=C:\WINDOWS
V:Path=...
V:HTTP_USER_AGENT=Opera/7.23 (Windows NT 5.1; U)  [en]
V:HTTP_HOST=mats2.gar.no
V:HTTP_ACCEPT=text/html, image/png, image/jpeg, image/gif, */*;q=0.1
V:HTTP_ACCEPT_LANGUAGE=no;q=1.0,en;q=0.9
V:HTTP_ACCEPT_CHARSET=iso-8859-1, utf-8, utf-16, *;q=0.1
V:HTTP_ACCEPT_ENCODING=deflate, gzip, x-gzip, identity, *;q=0
V:HTTP_REFERER=http://mats2.gar.no/cgi-bin/gweb?7800=tp8v2
V:HTTP_CONNECTION=Keep-Alive, TE
V:HTTP_TE=deflate, gzip, chunked, identity, trailers
V:CONTENT_TYPE=application/x-www-form-urlencoded
```

```
V:CONTENT_LENGTH=55
N:1
0:12,37,1,0
S:24,80

  <<< GALLAGHER & ROBERTSON - OPENING UP THE WORLD FROM YOUR DESKTOP! >>>


             Welcome to GAR-WS on System-J in Phoenix


                 GCOS8 release: SR5.1.1
                 TP8 Release:   8IT4.3


                 <XMIT> to continue  .



   This TP8 system is for demonstration only. Its purpose is to
   demonstrate GCOS connectivity from the Internet using various G&R
   products.
```

The external script interface: output data

The output interface defines the actions that your script would like Gweb to take. Your script simply prints data on the standard output, and Gweb analyzes whatever you print and acts accordingly.

The output must start with a line containing nothing but a number. This number defines what the rest of the output is.

Number	Script type	Output type	Description
2	Input and output scripts	Send data to the application	The rest of the output stream contains a CPI-C keycode sequence to be sent to the application, instead of sending HTML code to the browser.
3,13	Input and output scripts	Send HTML to the browser	The rest of the output stream contains HTML output that should be sent directly to the browser. Gweb will not change this in any way, which means that no `#macro#` expansions will occur. Code 3 includes the Gweb header and footer files. Code 13 suppresses them. In this case you are responsible for all HTML needed to set up the page, including `<HTML><HEAD>`... and `</BODY></HTML>`.
23,33	Input and output scripts	Send HTML to the browser	The script output can contain `#macro#` directives that are expanded by Gweb. Code 23 includes the Gweb header and footer files. Code 33 suppresses them.
43,53	Input and output scripts	Send HTML to the browser	The script output is a single line containing the name of a file that should be expanded by Gweb and sent to the browser. Code 43 includes the Gweb header and footer files. Code 53 suppresses them.

Number	Script type	Output type	Description
4	Output scripts only	Define content type	The next line in the output stream contains the MIME content type of the document sent to the web browser. Unless you know exactly what you're doing, do not specify this directive. The default content type/subtype is `text/html`. Typically, after these lines you send a "3" line followed by a number of lines with HTML output.
1001 - 1050	Input and output scripts	Set user-defined variables	The next single line in the output stream contains the value for User-supplied parameter number N-1000. For example, to set user-parameter number 16 to the value Gallagher, let your external script output the following: **1016** **Gallagher** This can be repeated to return several user parameters. To remove the value of a variable, simply output a blank line.
	Input and output scripts	(none of the above)	If the first line of the output does **not** contain any of the predefined values, the output is discarded. Gweb will continue normal operations, including header/footer files.

User-supplied parameters

You can supply external parameters to a Gweb session, there are a total of 50 user-specified parameters, numbered 1 thru 50:

```
gweb.cfg        -paramN ......
Gweb URL        <A HREF="/cgi-bin/gweb?paramN=....>
GET parameters  <A HREF="#gweb#?paramN=...&#get_data#
In a form       <input type=text name=paramN value="..".
```

They are passed to scripts as variables and returned in the output stream:

```
V:ParamNN=.........                          value of NN

10NN
.........                                    value of NN
```

You can return several user parameters simply by repeating the 10NN and value lines. These parameters must be returned before using output codes 2 (CPIC-sequence), 3|13 (HTML without parsing), 23|33 (HTML with parsing) or 43|53 (filename for HTML).

They can hold any text string, and can be used both as a facelifting macro (see page 100) and as CPIC keyboard code sequences (see below).

User parameter in CPIC keycode sequences

You can include user parameters within CPIC keycode sequences:

```
...@#param(N)#...
```

If user parameter number N is defined, the `@#param(N)#` string is replaced by the parameter value. If it is not defined, the `@#param(N)#` text is simply removed from the CPIC keyboard code sequence.

Print support

In the case of VIP7800 or DKU terminal emulation, it is possible for the mainframe application to send print data addressed to the printer 'attached' to the terminal.

The VIP and DKU emulations collect print into 'print jobs' and deliver the jobs to Gweb. The definition of a print 'job' depends on the configuration of the emulation, but the default is a timer that delivers print as a 'job' whenever there has been no further print for the duration of the timer. You may need to adjust the emulator configuration to receive Gweb print in a timely manner.

Gweb holds the print jobs until it is asked to print them by the browser. The browser asks for print using a field in a request in the same way as 'transmit', 'break' or 'disconnect'. The request is explained below.

The browser must know that print is available in order to request it, and this is accomplished using the facelifting macro that tests for print availability `#print(QUERY)#`. A test on print availability is included in the standard Gweb footer, so that every HTML page sent to the browser indicates if there is print available:

```
#if print(pending)#
#include file print_js.txt#
#elsif print(spooling)#
<P><A HREF="#gweb#?#get_data#&cmd=Refresh" #js(Href)#>Host
print spooling - refresh screen</A></P>
#endif#
```

In the case when there is a print job waiting, (`pending`) it includes the HTML file `print_js.txt`. This file can be replaced by `print_vb.txt` for printing 'transparent' data. This is explained below.

If there is no print finished, it tests if there is print currently being delivered (`spooling`), and if so, it includes a hyperlink with a command field, value `refresh` that, if clicked, causes Gweb to refresh the screen without any application interaction. The user, seeing the hyperlink, knows that there is print being delivered, and can choose to proceed normally with the screen dialog (receiving the print later when it is finished), or to refresh the screen.

Printing text only data

Text-only data consists of the normal printable character set sent by the application (by default transliterated by the emulation to ISO 8859-1). Additionally Gweb interprets CR (carriage return), LF (line feed) and FF (form feed).

Open a new window by JavaScript or HREF

The standard Gweb footer, if it detects that there is a print job, includes the file `print_js.txt`, the file content is:

```
<SCRIPT LANGUAGE="JavaScript"> <!--
var wMode='directories=0,location=0,status=0,titlebar=0,toolbar=0';
var wURL='#gweb#?print&#get_data#';

if (!isMSIE || "object" != typeof(window.open(wURL,'_blank',wMode))) {
   document.write('<P><STRONG>Process <A HREF="#gweb#?print&#get_data#"
   TARGET="_blank">host print<\/A><\/STRONG><\/P>');
   } // the window was not created. Display a hyperlink instead
// --> </SCRIPT>
```

This JavaScript code attempts to open a new window, with a URL that points to Gweb and contains a request to deliver the print data. If this is not successful, it includes a hyperlink instead, so that the user can click on it to initiate print. The print command uses the default encoding (text). All characters reserved for HTML syntax (" & < >) and high-ASCII are quoted according to the '&entity;' HTML standard. The line feed and form feed formatting information from the mainframe application will be preserved.

Please note that if your users have browsers or third party software that suppresses 'pop-up' windows, and additionally does not report back that the pop-up failed (Opera, for example, has an option that works like this), then the print will not be reported. In that case you must modify your Gweb footer, or `print_js`, to use a hyperlink only.

Display or print the window

On receiving the print request Gweb sends the print data, enclosed in HTML, and formatted according to the file specified in `-prt` in `gweb.cfg`. The default if there is no `-prt` is:

```
<html>
<head><title>#title# print</title></head>
<body><pre>
#print text#
</pre></body></html>
```

The default simply displays the print. It assumes that the print is being sent to a separate window.

We include a file `gwebprt.htm` in the Gweb directory that can be used to automatically print the print window and close it afterwards. The content is:

```
<html>

<!--
This example Gweb print file:
- includes a small JavaScript program that automatically displays the
  Print dialog box, and closes the window afterwards
- sets the title of Gweb's web pages using the <title> tag
-->

<head>
<SCRIPT LANGUAGE="JavaScript"> <!--
function PrintAndClose()
   {
   window.print();
   window.close();
   } /* PrintAndClose */
// --> </SCRIPT>

<STYLE TYPE="text/css" MEDIA="print"> <!--
 DIV { page-break-before: always; }
 PRE { font-size: 12pt; }
// -->
</STYLE>

<title>#title#</title>
</head>

<body onLoad="PrintAndClose()">
<pre>
#print text#
</pre></body></html>
```

The call to the JavaScript is included in the body tag and used when the browser loads the page. The JavaScript causes the browser to automatically open the printer selection box so that the user can choose a printer, and then prints the content of the window. Afterwards it closes the window.

Note that the font used for the print page can be varied to suit your printer by changing the style used in `gwebprt.htm` above. The page layout used by your browser for the `window.print` function, including headers, footers and margins can be set up under the ‘File’ menu entry in the browser’s menu bar.

Printing transparent print

Transparent print consists of binary data that includes printer control sequences, and can only be understood by the intended printer. Gweb must encode the binary data in a form that can be delivered to the browser, ands then start a routine locally on the browser that can decode the data again, and deliver it as binary data to the printer.

Change Gweb's standard footer

If you need to print transparent print data, you must modify Gweb's standard footer to include `print_vb.txt`. Modify it as follows:

```
#if print(pending)#
#include file print_vb.txt#
#elsif print(spooling)#
<P><A HREF="#gweb#?#get_data#&cmd=Refresh" #js(Href)#>Host
print spooling - refresh screen</A></P>
#endif#
```

In the case when there is a print job waiting, (`pending`) it includes the HTML file `print_vb.txt (see below).`

If there is no print finished, it tests if there is print currently being delivered (`spooling`), and if so, it includes a hyperlink with a command field, value `refresh` that, if clicked, causes Gweb to refresh the screen without any application interaction. The user, seeing the hyperlink, knows that there is print being delivered, and can choose to proceed normally with the screen dialog (receiving the print later when it is finished), or to refresh the screen.

Microsoft Internet explorer only

Unlike the JavaScript routines used above to print text only, the VBscript supplied with Gweb for printing transparent print data only works with MSIE.

The use of VBscripts that interact with the browser window, and deliver binary data to the printer represents a security risk. In order for this to work you must in your browser enter Tools => Internet Options => Security => Custom Level, and enable the items related to ActiveX controls (at least allow the prompt).

Embedded print, and VBscript

The content of `print_vb.txt` is:

```
<!-- Include file for printing with Visual Basic in
Microsoft Internet Explorer -->

<FORM NAME="PrintForm" value="
#print qpr#
"></FORM>

<SCRIPT language="vbscript">
function HexBin(s)
   dim n,r1,r2

   n = asc(mid(s,1,1))
   if (n > 57) then r1=n-55 else r1=n-48
   n = asc(mid(s,2,1))
   if (n > 57) then r2=n-55 else r2=n-48

   HexBin = chr ((r1 * 16) + r2)
end function

sub transparentPrint()
    dim sFname          ' File name
    dim fs              ' File system
    dim fil             ' File "handle"
    dim txt             ' HTML embedded text
    dim i               ' Index to scan the whole text
    dim c               ' Current character under processing

    sFname = InputBox ("Selection:", "Select your printer / file", "LPT1")
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set fil = fs.CreateTextFile(sFname, True)
    txt = document.PrintForm.value
    for i = 1 to len(txt)       ' Scan all
       c = mid(txt,i,1)         ' Current character
       if c >= " "  then        ' ignore controls inserted by Gweb
          if c = "=" then       ' Decode hex characters
             c = HexBin (mid (txt,i+1,2))
             i = i + 2
             end if
          fil.write(c)
       end if
       next
     fil.Close
end sub

transparentPrint()
</SCRIPT>
```

The macro `#print qpr#` embeds the print data into the HTML in a way invisible to the user. It is encoded as ‘printed quotable’ so that the browser can handle the binary data. The VBscript decodes the quoted printable data back into binary data, and delivers it character by character to a file, which is in fact the users printer (LPT1 by default).

JavaScript automation

Gweb implements terminal-like keyboard functionality and variable field validation on the client (browser) side by including comprehensive JavaScript routines in the HTML versions of the screens. Keyboard functionality such as auto-tab, and field validation is done on the fly, as you type. Further field validation is done upon submission of the HTML form from the browser.

Browser requirements

The JavaScript routines are qualified using Win32 Netscape version 6.2 and Win32 Internet Explorer version 6.0. Other browsers may provide limited functionality, but these are not qualified or guaranteed by G& R.

JavaScript activation

Gweb includes information needed by the JavaScripts, and the JavaScripts themselves in the standard header `gwebhead.htm`. The information, in the form of JavaScript tables always precedes the JavaScript methods.

```
<SCRIPT LANGUAGE="JavaScript"> <!--
#javascript(variables)#
// --> </SCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/gweb/gwebkbd.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="/gweb/gweb.js"></SCRIPT>
```

The macro `#javascript(variables)#` and files are described below. The files are delivered with Gweb, and should be placed in the `gweb` sub directory of the directory that is defined to be the web server DocumentRoot. See your web server documentation for configuration details.

Gweb activates the methods by inserting calls into the HTML tags used in the fields of the HTML (see ***Gweb configuration***, section ***JavaScript settings***) e.g.:

```
<body onLoad="doFocus()" onKeyDown="return doKeyDown(event);"
onUnload="doCleanup()" onBeforeUnload="doUnload()">

<input type="text" onKeyPress="return doKeyPress(event);" onKeyUp="return
doKeyUp(event);" onFocus="doSelect(this);">
```

The validation flags

Gweb provides the JavaScript methods with information, including information as to the number and type of variable fields. Gweb uses the macro `#javascript(variables)#` to include this information. An example is:

```
var closeWindow=0; if (1 == closeWindow) window.close();
var emulationType="7800";
var guard=true;
var cursorField="f0";
var sessionTimeout=14400;
var disconnectURL="/cgi-
bin/gweb?sid=8276.TPLXRJDE.2&command=Disconnect&closeWindow=1";
var scriptName="/cgi-bin/gweb";
var sessionId="sid=8276.TPLXRJDE.2";

  var fieldInfo = [ // fldCount=4
  { name:"f0",  maxlength:3,  check:"",   validity:0 },
  { name:"f1",  maxlength:3,  check:"E",  validity:0 },
  { name:"f2",  maxlength:3,  check:"F",  validity:0 },
  { name:"f3",  maxlength:2,  check:"+",  validity:4 }
  ];
```

"`fldCount=4`" (number of fields) is a comment only.

Each "`name`" key has a unique field name generated by Gweb as its value.

Each "`maxlength`" key specifies the maximum number of characters allowed in the field and is used to activate auto-tabbing.

Each "`validity`" key specifies allowable characters for each field. The key values are mapped to an allowable character set using a table included in the JavaScript methods. Currently they are as follows:

0	(Everything)	All printable characters are allowed.
1	Numeric dku	0123456789 #$,.-+£€
2	DKU/Vip77 GS4	0123456789:;<=>? !\"#$%&'()*+,-./£ €
3	3270 5250 VIP78	0123456789 ,.+-
4	Digits (0-9)	0123456789
5	Alphabetic	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz,.-
6	Wincom GS4	0123456789+,-./£€

Each "check" key has a string value that specifies one of:

""	do nothing	"F"	enforce ‘must fill’
"E"	enforce ‘must enter’	"+"	enforce ‘signed field’

The function key bindings

These are stored in the file gwebkbd.js and specify the key/modifier combinations for simulating the function and special purpose keys on a terminal. The bindings are included in a table in the format:

```
fKeyMap = [
 { keyCode:112,  isMapped:true, shift:false, ctrl:false, alt:false,
vKey:"cmd:F1"  }, // F1
 { keyCode:113,  isMapped:true, shift:false, ctrl:false, alt: false,
vKey:"cmd:F2"  }, // F2
...
...
 { keyCode:46,  isMapped:true, shift:false, ctrl:false, alt:true,
vKey:"ClearScreen" }, // Alt/Del
 { keyCode:36,  isMapped:true, shift:false, ctrl:false, alt:false,
vKey:"CursorHome" },  // Home
```

The first line of the table maps keycode 112 (normally generated by ‘F1’) to the Gweb command for Function key 1. The last line maps keycode 36 (normally the ‘Home’ key) to the JavaScript function CursorHome. Please refer to the file itself for complete documentation of the Gweb commands and JavaScript functions that are mapped on a standard keyboard.

Your keyboard may not generate the keycodes assumed in the table, or you may have your own preferences for which keyboard keys you will use for the various Gweb commands or JavaScript keyboard functions. In this case you can simply modify the table above, replacing the keyCode and modifier (shift, ctrl, alt) as you prefer.

To help you do this the Gweb delivery includes a HTML page that you can use to check the keycodes generated by your keyboard:

```
\gweb\gwebkbd.htm
```

This page will first test your browser’s version; to see if it qualified for use with the JavaScripts as delivered by G&R. It will warn you if your browser is not qualified.

Thereafter you can press any key on your keyboard, by itself or in combination with one or more qualifiers, to see what your browser delivers to the JavaScript used to activate the test page. You can then use this information to modify the keyboard-binding table.

The JavaScript methods

These are collected in the file `gweb.js`. Gweb automatically inserts JavaScript extensions in the generated HTML tags. These extensions call the appropriate JavaScript functions as defined in the configuration file e.g.:

```
<body onLoad="doFocus()" onKeyDown="return doKeyDown(event);"
onUnload="doCleanup()" onBeforeUnload="doUnload()">

<input type="text" onKeyPress="return doKeyPress(event);" onKeyUp="return
doKeyUp(event);" onFocus="doSelect(this);">
```

The JavaScript methods we supply are qualified only for Netscape and Internet Explorer on the Windows 32-bit platforms, and only for the releases stated in ***Browser requirements*** above.

Customization

Gweb JavaScript functionality is provided by G&R as a browser-enhancement that increases browser keyboard functionality and adds field validation functionality for our customers, provided they are using a qualified browser.

If you are using a qualified browser we recommend that you limit your customization of Gweb JavaScript to editing the function key bindings provided by `gwebkbd.js`. Making changes to the Gweb JavaScript methods we deliver in `gweb.js` requires a good grasp of JavaScript.

If you are using some other browser it may be possible to qualify it, or to obtain partial functionality by modifying the JavaScript we deliver.

G&R and their distributors accept no responsibility for making the JavaScript routines function with any other than the stated qualified browser versions.

G&R or a distributor may offer to modify the scripts for qualification with other browsers, or for changes in functionality. Such offers are entirely the responsibility of the party making the offer, and will normally be done as a separate development contract on a time and materials basis.

Troubleshooting

Enabling facelifter and emulation trace

To enable tracing, add the Gweb trace directive to the Gweb section of the `gweb.cfg` configuration file. Trace is otherwise off, unless turned on automatically by a critical error:

```
-gweb
  -trace N
```

where N is the sum of the following values:

Value	Description
0	Suppress automatic enabling of trace by critical errors
1	Enable Facelifter-level and emulation trace (`gw`**`E_NNN`**`.dbg`, see below)
2	Enable Request-level trace (`gwr_`**`NNN`**`.dbg`, see below)
4	Enable extended tracing
8	Keep trace files. Ensures that trace files are kept on disk. Normally, if Gweb has detected no problems, the trace file is automatically deleted.
16	Force trace entries to disk. Each trace file entry will be forced to disk as a single I/O operation, in order to maximize the amount of logging in case of crashes. *This option slows down Gweb operation dramatically - use with caution.*

For example, to enable Extended Facelifter tracing and keeping the files regardless of problems, add 1+4+8 and use `-trace 13` parameter.

Locating facelifter and emulation trace file

UNIX location:	`/usr/gar/debug/<scid>/gwE_NNN.dbg`
Windows location:	`C:\gar\debug\<scid>\gwE_NNN.dbg`

(<scid> = your DSA node name)
(E =one-letter emulation type; 3=3270, 5=5250, 7=7800 and d=dku)
(NNN = a number indicating a program instance)

This file contains internal details of Gweb execution, e.g. the full 24x80 screen image as received from the host and internal CPI-C commands executed, as well as internal information from the emulation.

Locating request trace files

UNIX location:	`/usr/gar/debug/<scid>/gwr_NNN.dbg`
Windows NT location:	`C:\gar\debug\<scid>\gwr_NNN.dbg`

(<scid> = your DSA node name)

Note that one file will be produced per request from the browser (i.e. each time you press the Transmit button). Long sessions with Request tracing enabled can produce (literally) thousands of trace files.

Line-level log

UNIX location:	`/usr/gar/debug/XXX/gwE_NNN.gli`
Windows NT location:	`C:\gar\debug\XXX\gwE_NNN.gli`

(XXX = user name; usually "gar" on Unix and "system" on Windows NT)
(E =one-letter emulation type; 3=3270, 5=5250, 7=7800 and d=dku)
(NNN = a number indicating a program instance)

This contains information from the line handler. You enable it in the line section of the `gweb.cfg` configuration file. Use -s_ for a session trace of the interchanges between the line module and the mainframe. Use -d_ for a data trace of the interchanges between Gweb's emulation and the line module:

```
-li XXX
   -s_ on
   -d_ on
```

Appendix: CPI-C keyboard codes

The CPI-C keyboard codes are a way of specifying the various key and function codes you can deliver to a given terminal emulation. The '@' character is used as an escape key followed by a mnemonic code that corresponds to the supported emulation function. An example is Transmit, which is coded as `@E`. Please see the *G&R/CPI-C* manual for details.

NOTE: If you want to use the 'commercial at' sign '@' itself, you must use the two-byte code '@@'.

Keyboard codes for all emulations

Function	Code
@	`@@`
Gweb macros and user supplied parameters	`@#...#`
Backspace	`@<`
Backtab (Left Tab)	`@B`
Clear	`@C`
Cursor Down	`@V`
Cursor Home	`@0`
Cursor Left	`@L`
Cursor Right	`@Z`
Cursor Up	`@U`
Delete Character	`@D`
Host Break (3270/5250: "ATTN")	`@A@H`
Duplicate Character	`@S@X`

Function	Code
Transmit Data (3270/5250: "Enter")	`@E`
Erase EOF	`@F`
Erase EOP	`@G`
Insert Mode	`@I`
Left Tab (Back Tab)	`@B`
New Line	`@N`
Reset	`@R`
Right Tab (Tab)	`@T`
Refresh screen	`@W`
Disconnect	`@Q`
F1/PF1	`@1`
F2/PF2	`@2`
F3/PF3	`@3`
F4/PF4	`@4`
F5/PF5	`@5`
F6/PF6	`@6`
F7/PF7	`@7`
F8/PF8	`@8`
F9/PF9	`@9`
F10/PF10	`@a`
F11/PF11	`@b`
F12/PF12	`@c`

Specific 7800 keyboard codes

Function	Code
Set Function Code Character: 'c'	@A@C@c
Attribute: 'c'	@S@Ac
Delete Attribute	@S@B
Delete Line	@S@D
Insert Line	@S@I
Previous Segment in 72 line mode	@S@M
Next Segment in 72 line mode	@S@N
Paste Line	@S@P
Reset Initial State	@S@R
Tab Clear	@S@c
Fold Line	@S@f
Unfold Line	@S@u
Carriage Return	@S@r
Skip Line	@S@s
Tab Set	@S@t
Transmit All	@P
Insert Mode Reset	@O
Shifted F1	@d
Shifted F2	@e
Shifted F3	@f
Shifted F4	@g
Shifted F5	@h
Shifted F6	@i
Shifted F7	@j
Shifted F8	@k

Function	Code
Shifted F9	`@l`
Shifted F10	`@m`
Shifted F11	`@n`
Shifted F12	`@o`

Specific DKU keyboard codes

DKU Function key mapping

Function	Code
FKC-0 Enter FKC mode and send only FKC string	`@A@A@0`
FKC-1 Enter FKC mode, send FKC string and screen	`@A@A@1`
When in FKC mode enter ASCII controls as hex ‘hh’	`@Xhh`
Terminate FKC mode and transmit	`@E`
Set Function Code Character: 'c'	`@A@C@c`

A common requirement in the DKU environment is mapping of the function keys to set the ‘function codes’ in the VIP-header (FC1, FC2) and/or define FKC sequences that are transmitted when the key is pressed. The FKC sequence is optionally followed by the content of the screen, depending on the FKC mode e.g.

Mapping F1 to set FC1 to 'a', but transmit no FKC sequence:

```
-f1 @A@C@a@A@A@0@E
```

Mapping F1 to transmit the standard sequence, followed by the screen:

```
-f1 @A@A@1@X1b[1u@E
```

DKU specific control keys

Function	Code
Clear All Tab Stops	`@S@a`
Back Character	`@S@b`
Delete Line	`@S@D`
Insert Line	`@S@I`
Paste Line	`@S@P`
Clear Tab Stop	`@S@c`
Fold Line	`@S@f`
Unfold Line	`@S@u`
Carriage Return	`@S@r`
Skip Line	`@S@s`
Set Tab Stop	`@S@t`
Insert Character	`@S@i`
Paste Character	`@S@p`
Transmit Page	`@P`

Specific 3270/5250 keyboard codes

Function	Code
System Request ("SYSREQ")	`@S@S`
Cursor Select	`@A@J`
Field Mark	`@S@y`
PF13	`@d`
PF14	`@e`
PF15	`@f`
PF16	`@g`
PF17	`@h`
PF18	`@i`
PF19	`@j`
PF20	`@k`
PF21	`@l`
PF22	`@m`
PF23	`@n`
PF24	`@o`
PA1	`@x`
PA2	`@y`
PA3	`@z`

Specific 5250 keyboard codes

Function	Code
Test	@S@T
Field +	@+
Field -	@-
Field Exit	@X
Roll Down	@S@M
Roll Up	@S@N
Help	@H

Appendix: Host Links Manuals

Below you find a complete list of all available Host Links manuals:

Installation	
Host Links Servers	Installation and Configuration on UNIX/Linux
Host Links Emulators	Installation and Configuration on UNIX/Linux
Host Links	Installation and Configuration on Windows
Line handling	
Gline	Line Handler and DSA/OSI Configuration
Ggate	Transparent Gateway
Gproxy	Network Manager & SNMP Proxy Agent
G&R SSL	Using SSL for security in G&R products
GlAPI	Application Programming Interfaces
Gsftp	Gateway between FTP and SFTP
Emulations	
Gspool	Network Printer Emulation
GUFT	Unified File Transfer
G3270	Emulating IBM 3270 Terminals
G5250	Emulating IBM 5250 Terminals
Pthru	Gateway to the Bull Primary Network
Qsim	Emulating Questar DKU7107-7211 & VIP7700-7760
V78sim	Emulating VIP7801 & VIP7814
Gweb	Web Browser Front-end for DKU, VIP7700-7760, VIP7800, IBM3270 and IBM5250 Emulations

Appendix: Host Links DSA Utilities

The Gline package includes a set of Gline communication utilities. These are used when testing and debugging connection problems. The utilities are delivered as part of the Gline package and can be used without any additional configuration. The nodes to be tested must of course be configured in the `dsa.cfg` file.

Gconame

Lists the parameters generated from a given CONAME. The utility works for both CONAME and RESOURCE e.g.:

```
gconame tnviptm

Checking 'dsa.cfg' for coname 'tnviptm'
Coname: tnviptm, type TM, parameters:
-DA misfld
-S_
-D_
-CODE 0000
-CODE 1000
-CODE 1800
-TEXT Remote SCID?:
-CODE 4700
-TEXT Remote application?:
-CODE 1400
-CODE 1600
-TEXT Password?:
```

Gerror

Shows the text message associated with a DSA reason code. Only the most common codes are supported i.e. the ones related to network, transport and session communication layers. Errors generated by the OSI-stack on the Host Links platform are not covered by this utility; please refer to the documentation from the vendor of the stack e.g.:

```
gerror 0109
Reporting component: Session control (01) 0109, Dialog
protocol error or negotiation failed (wrong logical record).
```

For a detailed description of all reason codes, please consult the Bull manual ***OSI/DSA Network System Messages and Return codes*** (39A2 26DM).

Glnode

List and verify the communications parameters of the local node e.g.:

```
glnode
Local node name : GRDL
Local session control id : GRDL
DSA200 address (area:tsm): 54:60 (36:3C)
```

Gmacfix

When you connect to FCP cards on Bull mainframes via an Ethernet port on the LAN-Extender the mainframe address is given in Ethernet (LLC) format. If you connect to an FDDI adapter you must convert the MAC address to SMT. e.g.:

```
gmacfix 080038000fab
MAC address 080038000fab = 10001c00f0d5
```

Gping

Connects to a remote system using the Gline parameters set on the command line. If successful it returns ‘connected to application’, otherwise it shows the error code returned e.g.:

```
gping -li dsa -dn b7dl -da iof -du jim -pw mydogsname
Gping - $$DSA: Connected to application
```

Grnode

Return the parameters (in `dsa.cfg`) and the state of a remote node e.g.:

```
grnode b6dl
Checking 'dsa.cfg' for node 'b6dl'
Session control id : B6DL
DSA200 address (area:tsm) : 1:5 (1:5)
Inactivity interval : 0
Route 0
Load balance percentage : 0
TP class : 2
TP expedited : 0
TPDU size : 0
Network address : 130405
```

Gtrace

Same as `gping` but writes the DSA communication trace on the user's terminal (applicable to UNIX versions) e.g.:

```
gtrace -li dsa -dn ln40 -da snm151
D6:Application event @ 14:17:17.6003. tokenitem = 00
D6:Application event @ 14:17:17.6082. tokenitem = 00
D6:Connect request called, node = LN40
D6:OurBufferSizes. ApplMaxXmit = 511, ApplMaxRecv = 500
Rec:4000 0002 s:2
Rec:506B 0010 s:16
etc etc
Gtrace - line trace ending.
Gtrace - $$DSA: Connected to application.
```

Gtsupd

Update the state of a transport route. Transport routes can be set automatically in a disabled state if a backup route is configured. When such a state change occurs the route will be set back to the enabled state after a configurable timer has expired. The default is 15 minutes. You can reset the state of such a route with `gtsupd ts-name enbl/used/down/locked` e.g.:

```
gtsupd gars_rfc enbl
TS-entry 'gars_rfc', new state = enbl
```

Appendix: Host Links Trace

If you experience any kind of problem when using a Host Links application, the application trace file and/or the line handler trace file will provide useful documentation of the problem.

Trace activation

The Host Links products automatically create sub-directories in the debug directory when debug is activated: at product level using the `-dbg` parameter, or at line level using the `-d_` or `-s_` parameters to the line module.

Windows server	`gspool -id gs1 –dbg –ps \\SERVER\LEXMARK -li dsa -da tptst -d_ on`
UNIX Linux	`gspool -id gs1 -dbg-pc lp -li dsa -da tptst -d_ on`

Most G&R products include a facility for setting product or line parameters dynamically. It is therefore generally possible to turn on debug or trace without modifying the command line or configuration of a production system.

Trace types

All Host Links products accept a parameter `–dbg,` which starts an application level trace of internal events. This is useful when investigating malfunctions or looking closely at product behaviour.

All Gline line handlers accept a parameter `-d_` to turn on a data trace. It records data and enclosure level being exchanged with the line handler. It is useful when documenting product malfunction e.g. an emulation error, because it records exactly what the host sends and what the G&R application replies. It can be used to simulate a customer situation, reproduce a problem and to verify that a correction fixes the documented problem.

All Gline line handlers accept a parameter `-s_` to turn on a session trace. It records the raw data being exchanged between the line module and the underlying transport layer (e.g. OSI Transport, or TCP socket), as well as internal events and protocol states. It is useful when investigating protocol failures such as unsuccessful connect attempts or abnormal disconnections.

Structure

The Host Links file structure includes a debug directory to collect the trace and debug files in one location where the permissions can be adjusted as required for security. By default only the Host Links administrator can access the directory. The debug directory is created by the initialization procedure and located (by default) in:

Windows server	`\gar\debug`
UNIX Linux	`/usr/gar/debug`

If the application is a client type of application, a debug sub-directory with the same name as the user (UNIX username or PC login name) is created and all debug files are located there. This includes the line level trace except in the special case where the client application connects via Ggate and the line level trace is written on the Ggate system using the Ggate DSA node name as a debug sub-directory.

If the application is a server type of application, then a sub-directory will be created using the DSA node name on behalf of which the server application is executing. If the server does not use DSA the default local session control name is still used if there is a `dsa.cfg` file. If there is no `dsa.cfg` file then the system's UNIX or Windows communications node name is used. You can find this name using the command `uname –n` on UNIX systems, or the Network section of the control panel on Windows systems. This covers situations where several instances of a server are executing on the same system and accepting incoming calls to different DSA node names, or where several Host Links systems using the same server product share a file system.

Tracing Ggate

When Glink, a Host Links client or a customer application based on GlAPI connects through Ggate to the application, the line handler trace is generated on the Ggate system, with the name and location shown in the table:

Windows server	`\gar\debug\NODE\ggaNN-PPPP.dbg`
UNIX Linux	`/usr/gar/debug/NODE/ggaNN-PPPP.dbg`

NODE is the local DSA node name used by the Ggate system.

The trace file name consists of the prefix `ggaNN-` followed by the IP-address of the client, suffixed by `.dbg` for a terminal session or `–dbp` for a printer session. The following is a trace file name for Ggate session sequence number 5 executing on Host Links system `GRDL` initiated from a Glink client on IP-address `jim.gar.no:`

```
gga05-jim.gar.no.dbg
```

This file, and possibly also a Glink debug file and a Glink communication trace file activated by the `/J` command line parameter will be needed by the support engineer investigating any problem.

To enable a line handler trace through Ggate the product's start-up command or configuration file would look like this:

```
-LI YYY:ZZZZ -S_ -D_
```

(YYY =line handler identification, i.e. DSA or DIWS)
(ZZZZ =IP-address of the system running Ggate)

Examples - G&R products

Examples of directory and file names in the debug structure are:

/usr/gar/debug/jim	**Debug directory for user 'jim'**	
`qsm.dbg`	Qsim emulator debug file	`-dbg`

`qsm-gli.dbg`	Qsim host line trace	`-li dsa -s_`
`pth-glit.dbg`	Pthru terminal line trace	`-term -s_`
`pth-glih.dbg`	Pthru -host line trace	`-li dsa -s_`
`g32.dbg`	G3270 emulator debug file	`-dbg`
`g32-gli.dbg`	G3270 host line trace	`-s_`
`/usr/gar/debug/mike`	**Debug directory for user ‘mike’**	
`v78.dbg`	V78sim emulator debug file	`-dbg`
`v78-gli.dbg`	V78sim host line trace	`-li dsa -s_`
`guf.dbg`	GUFT client debug file	`-dbg`
`guf-gli.dbg`	GUFT client host line trace	`-li dsa -s_`
`/usr/gar/debug/en01`	**Debug directory for node ‘en01’**	
`guf.def`	GUFT server debug file	`-dbg`
`guf-gli.def`	GUFT server host line trace	`-li dsa -s_`
`gli-gli.dsa`	DSA listener host line trace	`-s_`
`gli-gli.diw`	DIWS listener host line trace	`-s_`
`gsp.def`	Gspool (default `-id`) debug file	`-dbg`
`gsp-gli.def`	Gspool (default `-id`) host trace	`-li dsa -s_`
`gga01-mike.gar.no.dbg`	Ggate line trace, first Glink	`-s_`
`gga02-mike.gar.no.dbg`	Ggate line trace second Glink	`-s_`
`/usr/gar/debug/en02`	**Debug directory for node ‘en02’**	
`gsp.abc`	Gspool (-id `abc`) debug file	`-dbg`
`gsp-gli.abc`	Gspool (-id `abc`) host trace	`-li dsa -s_`
`gspc-gli.def`	Gspool DPF8 command trace	`-li tcp -s_`
`gspd-gli.def`	Gspool DPS8 data trace	`-li tcp -s_`

`gsp._00`	Gspool started on demand debug	`-dbg`
`gsp-gli._00`	Gspool started on demand trace	`-li dsa -s_`

CPI-C and Gweb trace files

Gweb uses the CPI-C libraries so the Gweb debug structure is exactly the same as for CPI-C, except that Gweb inserts its own product identifier into the file name structure. CPI-C applications use the 'client' style of debug and create a debug directory with the UNIX username or PC login name used by the process that started them.

The application level debug (-dbg) and line trace (-s_ and `-d_`) are set in the `cpic.cfg` file. The line trace goes to the debug directory, with the name built up as follows:

```
<product_id><session_id>-<process_id>.<debug_type>
```

product_id	*Value*	*Comment*
	`cpi`	CPI-C API
	`cp3`	CPI-C 3270
	`cp7`	CPI-C 7800
	`cpd`	CPI-C DKU
	`gw3`	Gweb3270
	`gw7`	Gweb7800
	`gwd`	Gwebdku
session_id	`(nn)`	If multi-session application, 1-63
process_id	`n (n n n...)`	Varies by platform
debug_type	`dgb`	Application level debug
	`gli`	Line trace

Example:

`\gar\debug\system`	debug directory for user `"system"`	
`cpi-16.dbg`	CPI-C single session debug	`-dbg`
`cpi-16.gli`	CPI-C single session line trace	`-li dsa -s_`
`cpi2-123.dbg`	CPI-C session 2 application debug	`-dbg`
`gw7-20172.gli`	Gweb7800 host line trace	`-li dsa -s_`

Appendix: Error Codes

OSI/DSA error codes

Below is a list of OSI/DSA error codes and the corresponding description. These are the same descriptions that the `G&R/Gerror` utility will display when given the DSA code as a parameter.

code	Description
00xx	**General Errors**
0001	Open Failure in LC - Reject for unknown reason
0002	Open Failure in LC - Acceptor customer node inoperable
0003	Open Failure in LC - Acceptor customer node saturated.
0004	Open Failure in LC - Acceptor mailbox unknown.
0005	Open Failure in LC - Acceptor mailbox inoperable.
0006	Open Failure in LC - Acceptor mailbox saturated.
0007	Open Failure in LC - Acceptor application program saturated
0008	Connection refused. Transport protocol error or negotiation failed.
0009	Open Failure in LC - Dialog protocol error or negotiation failed
000A	Open Failure in LC - Presentation protocol error or negotiation failed
000B	Open Failure in LC / Connection refused lack of system resources
000C	Open Failure in LC / Connection refused from GCOS7 duplicate user
000D	Open Failure in LC, Duplicate implicit LID / Q class not started
000E	Open Failure in LC, Duplicate GRTS Id / lack of memory resources
000F	Open Failure in LC, No Logical line declared for DACQ / 7 connection refused
0010	Open Failure in LC, GCOS 8 GW Missing translation / Incorrect device length in ILCRL.
0011	Open Failure in LC, DAC connection not initialized / Too many jobs executing
0012	Open Failure in LC, No binary transfer / impossible to start the IOF job
0013	Open Failure in LC, connection is not negotiated in FD mode / impossible to start the IOF job

0014	Disconnection - Timeout resulting from absence of traffic.
0016	Option missing for an RBF mailbox.
0017	Connection refused - Incorrect access right for MB.
0018	Connection refused - Incorrect access rights for the application.
0019	Connection refused - Unknown pre-negotiated message path
001A	Connection refused - Security validation failed.
001B	Connection refused - Unknown acceptor mailbox extension.
001C	Connection refused - Inoperable acceptor mailbox extension.
001D	Connection refused - Invalid Message group number.
001F	Disconnection - no more memory space.
0020	Connection refused - Unknown node.
0021	Connection refused - inaccessible node or Host down.
0022	Connection refused - saturated site.
0023	Connection refused - inoperable mailbox.
0024	(X.25) Packet too long. Problem with packet size. / Connection block already used.
0030	Syntax Error - option not known (received on close VC).
0031	(X.25) No response to call request packet - timer expired.
0033	(X.25) Timer expired for reset or clear indication.
0039	Disconnection - transport protocol error (MUX).
003C	Presentation Control Protocol Error
003E	The application has not the turn
003F	Message group closed
0040	(X.25) Facility code not allowed. / Connection refused - unknown node
0041	Connection refused - path not available.
0042	Connection refused - Duplicate USER ID / Facility parameter not allowed
0044	(X.25) Invalid calling address.
0045	(X.25) Invalid facility length.
0047	(X.25) No logical channel available.
004F	DNSC: (X.25) Invalid call packet length.
0050	Normal disconnection (GCOS3/8)
0051	Error or Event on LC initiated by GW
0052	Error or Event on LC initiated by GW.
0053	Error or Event on LC initiated by GW. TCall
0054	Error or Event on LC initiated by GW. DIA in LOCK State
0055	Error or Event on LC initiated by GW. DIA error
0056	Error or Event on LC initiated by GW. GW has no known explanation.
0057	Error or Event on LC initiated by GW. Reject mailbox permanent

0058	Error or Event on LC initiated by GW. No more input lines in DACQ
0059	Time-out on GCOS 3/8 gateway.
005A	Error or Event on LC initiated by GW. Disconnect from terminal without reason
005B	Error or Event on LC initiated by GW. Wrong letter or wrong record
005C	Error or Event on LC initiated by GW. Forbidden letter received
005D	Error or Event on LC initiated by GW. Forbidden letter received
005E	Error or Event on LC initiated by GW. No buffer for secondary letter
005F	Error or Event on LC initiated by GW. No buffer for fragmented letter
0060	Error or Event on LC initiated by GW. Disconnect on end of phase record
0061	Error or event on LC initiated by GW. No buffer for control letter.
0062	Error or event on LC initiated by GW. Mailbox in closing phase
0064	Error or event on LC initiated by GW. Flow control error.
0065	Error or event on LC initiated by GW. CH locked by operator.
0066	Error or event on LC initiated by GW. Disconnect with a normal TMG F2 exchange.
0067	Error or event on LC initiated by GW. Teletel rerouting error from DACQ
0068	Error or event on LC initiated by GW. Teletel routing error from DACQ
0069	Error or event on LC initiated by GW. Teletel rerouting error from TM
006A	Error or event on LC initiated by GW. Teletel rerouting error from TM
006B	Syntax error - text too long.
006C	Syntax error - illegal object in a GA command.
006D	Syntax error - unknown node Id.
0078	Syntax error - illegal command for this object.
0079	Syntax error - illegal date.
007F	(X.25) No route available for X.25 switching.
0081	No more network routes available for switching.
0082	(X.25) Hop count reached for X.25 switching.
0083	(X.25) Flow control negotiation error.
0085	(X.25) Frame level disconnection.
0086	(X.25) Frame level connection.
0087	(X.25) Frame level reset.
0090	Frame level not set.
0092	(X.25) X.25 Echo service in use.
0093	(X.25) Incorrect password for PAD connection.

0094	(X.25) No more PAD connections allowed.
0096	(X.25) TS SX25 or NU X25 objects locked.
009C	(X.25) Invalid packet header. X.25 protocol error.
009D	(X.25) Incompatible header. X.25 protocol error.
009E	(X.25) Logical Channel Number too high.
009F	(X.25) Incorrect packet type.
00B2	Use of invalid password through PAD
00B6	Unknown mailbox selection for PAD connection using the PAD password.
00C0	(X.25) Normal disconnection.
00D7	(X.25) TS image (of type DSA or DIWS) in LOCK state.
00DE	(X.25) NS RMT or NR SW in LOCK state.
00E1	Connection refused. Mailbox is not in ENBL state.
00E6	QOS not available permanently.
01xx	**Session Control**
0100	Logical connection accepted or normal termination
0101	Rejection for unknown reason or abnormal termination
0102	Acceptor node inoperable.
0103	Acceptor node saturated. When a node has no available resources
0104	Acceptor mailbox unknown.
0105	Acceptor mailbox inoperable.
0106	DNS: Acceptor mailbox saturated.
0107	DNS: Acceptor application program saturated.
0108	Transport protocol error or negotiation failed (DSA 200 only).
0109	Dialog protocol error or negotiation failed. (Wrong logical record).
010A	Time-out on session initiation / unknown LID
010B	Acceptor mailbox extension unknown.
010C	Acceptor mailbox extension inoperable.
010D	Invalid Session Number.
010E	Unknown node.
010F	System error. System generation error or insufficient memory space
0110	Application abnormal termination. Subsequent to an abnormal occurrence in the dialogue
0111	Normal terminate rejected.
0112	Protocol not supported.
0113	Session control service purged by user.
0115	Disconnection Time-out on message group initiation.
0117	Incorrect Access Right for MB
0118	Incorrect Access Right for the Application
0119	Pre-negotiated Message Path Descriptor unknown
011A	Security validation failed
011E	Incorrect object status

011F	Not enough memory space available.
0120	Node unknown.
0121	The channel object (CH) is in LOCK state
0122	Saturation - no plug available
0123	Object status = LOCK
0124	Connection block (TSCNX) already used
0125	Disconnection already running
0126	The connection block (TSCNX) is disconnected (or not connected)
0127	Change Credit value < 0
0128	Ineffective Change Credit (delta = 0)
0129	No more deferred letters
012B	"Reinitialization" Request
012C	"Reinitialization" in progress
012D	"Reinitialization" in progress, letters are dropped
012E	Close virtual circuit. Either no mapping exists between PA/NR or CL and VC/NS
012F	Null connection object index.
0130	Undefined function at Sysgen time.
0131	Letter too large with respect to the negotiated size.
0132	The received letter is longer than the size which was
0133	Disconnection of the session control user
0134	Interface error on EOR (End-Of-Record) processing.
013C	Presentation control protocol error.
013E	You do not have the turn.
013F	Message group closed.
0140	Session is closed.
0151	Request refused, no system buffers available.
0152	Incorrect addressing record.
0153	No presentation record in the ILCAL or ILCRL
0154	Negotiation failed on session mode
0156	Negotiation failed on resynchronization.
0157	Negotiation failed on END to END ACK
0158	No presentation record in the connection letter
0159	Negotiation failed on session mode
015A	Negotiation failed on letter size (in the Logical Connection record).
015B	Negotiation failed on resynchronization (in the Logical Connection record).
015C	Negotiation failed on end-to-end ACK (Logical Connection record).
015D	No support of the "letter" interface because Multirecord is not negotiated.
0160	Incorrect TSPACNX table.
0161	Protocol error on letter reception.

0162	Negotiation failure.
0163	Record header length error.
0164	Protocol error.
0165	Protocol error reception of control letter.
0166	Type or length error on interrupt letter.
0167	Protocol error on reception of data letter.
0168	Dialog protocol error.
0169	Unknown event.
016A	Protocol error on data transfer.
016B	Invalid status for a disconnection request.
016C	Invalid status for a recover
016D	Invalid status for a suspend/resume request.
016E	Negotiation failure.
016F	Unknown command.
0170	Error in presentation protocol
0171	Letter header length error in
0172	ILCAL is not DSA 200 protocol.
0173	Error in session record.
0174	Normal disconnection, without complementary reason code.
0175	Letter is not in ASCII or EBCD.
0176	Connection protocol letter header
0177	Letter header protocol error.
0178	Record header protocol error.
0179	Record header length error.
017A	Mbx record header length error.
017B	Error on buffer transfer.
017C	DSA 200 record header protocol
017D	DSA 300 record header protocol
017E	Unsupported connection options.
017F	Character error in ASCII string.
0180	No segmented record size.
0181	Invalid mailbox object index.
0182	Mapping error for a remote connection.
0190	No more buffers.
0191	Byte count is greater than GP.
0192	Byte count is greater than GP.
0193	Byte count is greater than GP.
0194	Byte count is greater than GP.
0195	Byte count is greater than GP.
0196	Byte count is greater than GP.
0197	Byte count is greater than GP.
0198	No more buffers.

0199	Byte count is greater than GP.
019A	Byte count is greater than GP.
019B	Byte count is greater than GP.
019C	Byte count is greater than GP.
019D	Byte count is greater than GP.
019E	Byte count is greater than GP.
019F	Byte count is greater than GP.
01A0	Invalid transfer state.
01A1	Suspend protocol running.
01A2	Suspend protocol running.
01A3	Recover protocol running.
01A4	Forbidden function in write request. ($WRITE)
01A5	Conflicting parameters for segmented record. (SWBREC)
01A6	Protocol conflict - suspend/recover.
01A7	Protocol not supported - letter/end-to-end ACK. (SWBLET)
01A8	Multi-record letter in progress.
01A9	Interrupt request forbidden.
01AA	Send control record request forbidden. (SCTROL)
01AB	Forbidden for TWA session - turn is here. (SREAD)
01AC	Termination forbidden - suspend or recover in progress. (STERM)
01C0	No space available for downstream connection request. (SMECNX)
01C1	No space available for upstream connection request. (SMUCNX)
01C2	No space available for upstream SCF connection. (SMRCNX)
01C3	No space available for session context. ($SCTX)
01E0	Enclosure or data length error for a write request. ($WRITE)
01E1	Enclosure or data length error for a write segment record request. (SWBREC)
01E2	Enclosure error for 'give turn' request. (SGVTRN)
01E3	Interrupt request is not demand turn, attention/data attention, or purge record.
01E4	Input status for a send control letter is not permitted.
01E8	Write request without turn.
01E9	Write segmented record request without turn.
01EA	Write segmented letter request without turn.
01EB	Send control letter request without turn.
01EC	Disconnection request without turn.
02xx	**Presentation Control**
0201	Protocol level not supported
0202	Application designation protocol error.
0203	Character encoding error. TM cannot support the proposed encoding.
0204	Character set error. TM cannot support the proposed character set.

0205	Character subset error. TM cannot support the proposed character subset.
0206	Incorrect record encoding.
0207	Incorrect parameter encoding.
0230	Data presentation control error. The presentation control proposed for this session cannot be used
0231	Device type is incompatible with the configuration.
0232	TM control protocol is incorrect.
0233	Device-sharing attributes are invalid.
0234	Initiator or acceptor configuration is not correct.
0235	Logical device index error.
0236	Number of logical devices is incompatible with the configuration.
0237	TM protocol record not supported.
03xx	**Terminal Management**
0300	Sysgen error WARNING. There is no mapped object; some objects will be spare.
0301	Operator requested session abort or logged.
0302	Idle time run out after secondary network failure.
0303	Idle time run out for no traffic.
0304	Form not found.
0305	Operator requested suspension.
0306	Destructive attention send on the session.
0307	Unknown TX addressed in this session. TM is unable to a the session.
030A	Protocol error. A record was received which did not comply with current standards
0310	Insufficient resources. The receiver cannot act on the request because of a temporary
031E	Incorrect value for Retry or Wait parameters on UP LL command.
0320	Function not supported.
0321	Parameter error. This can result
0322	Resource not available. The
0323	Intervention required (on principal device).
0324	Request not executable.
0325	EOI required.
0326	Presentation space altered, request executed.
0327	Presentation space altered, request not executed.
0328	Presentation space integrity lost.
0329	Device busy. The device is busy and cannot execute the request.
032A	Device disconnected.
032B	Resource not configured.
032C	Symbol set not loaded.

032D	Read partition state error.
032E	Page overflow.
0330	Subsidiary device temporarily not available.
0331	Intervention required at subsidiary device.
0332	Request not executable because of subsidiary device.
0340	TM cannot accept a new connection.
0341	Object status incorrect.
0342	The TM configuration is not correct.
0343	Unknown TX addressed on this session.
0344	Data presentation protocol error.
0345	Device type is incompatible with the configuration, or is not supported.
0346	TM control protocol incorrect.
0347	Device shareability attributes are invalid.
0348	Initiator or acceptor configuration is not correct.
0349	Logical device index error.
034A	Number of logical devices incompatible with the configuration.
0350	Disconnection of TM after reinitialization of the network.
0360	File not found. (Welcome and Broadcast Messages)
0361	Site not found. (Welcome and Broadcast Messages)
0362	NASF error. (Welcome and Broadcast Messages)
0370	No-session timeout. Device disconnected.
0371	No-input timeout. Device disconnected.
0372	No-output timeout. Device disconnected.
0373	Timeout due to no backup session being initiated.
0374	Timeout due to no backup session being established.
0375	Connection refused because of late activation of back up session.
0376	Disconnection of current session to switch to backup session.
0380	AUTOCN parameter not declared.
0381	Mixed ETB in data sent by VIP screen and cassette
0382	Data header sent by the terminal incorrect.
0383	Desynchronization in the exchange of data.
0384	KDS block count error.
038C	Remote terminal is not connected
0390	Unknown mailbox.
0391	No call packet to return.
0392	No "Possibility" command to return Protocol error
03C0	Slave device disconnection.
17xx	**Network Layer**
1701	PAD connection refused.
1702	Flow control error.

1706	Logical channel number not zero in restart packet.
1707	Illegal packet length or use of D-bit forbidden.
1708	Illegal header.
1709	Illegal Logical Channel Number.
1710	Invalid packet type for the automaton state. Protocol error
1711	Incorrect packet type.
1712	Inconsistent network parameters in the generation file.
1713	No more space.
1714	DSAC network layer object not usable.
1717	USED/ENBL transition. Transport station is locked.
1718	USED/ENBL transition. This is a back-up NR.
1719	USED/ENBL transition. Dynamic close due to load.
171A	USED/ENBL transition. Transfer time-out has elapsed.
171B	USED/ENBL transition. This is a back-up NR.
171C	USED/ENBL transition. Transport station is idle.
171E	USED/ENBL transition. NR object is locked.
171F	ENBL/LOCK transition. NR HDLC has no more memory space.
1721	Remote station is inaccessible via the configured network. Check
1723	Incorrect PAD password.
1724	Virtual circuit already in use. LCN (Logical Channel Number) too high.
1725	Invalid virtual circuit.
1726	Packet too short. Protocol error for the equipment directly connected to the Bull Datanet.
1727	Incompatibility between the generation parameters of two communicating systems on window or packet size.
1729	Packet size in communicating systems not the same.
1731	Timer runs out while waiting for call confirmation.
1732	Timer runs out while waiting for clear confirmation.
1733	Timer has run out while waiting a reset confirm.
1740	Call setup or call clearing problem.
1741	Open failure on virtual circuit. No flow control on this NS.
1742	Incorrect facility. Protocol error for the equipment directly connected to the Bull Datanet.
1744	Unknown subscriber.
1745	End of time-out on reset confirm. Invalid facility length. Protocol error for the equipment directly
1747	No logical channel available.
1749	End of time-out on call confirm.
174F	Incorrect packet length. Protocol error for the equipment directly connected to the Bull Datanet.
1755	Flow control, window, packet size or reset error.

1760	Frame disconnection.
1770	Frame connection.
1771	Frame reset.
1781	No more network routes available for X.25 switching.
1782	Maximum of 15 switches have been used,
1783	Flow control negotiation error.
1785	Frame level disconnection.
1786	Frame level connection.
1787	Frame level reset.
1790	Frame level not established.
1791	No more logical paths available for the PAD.
1792	Echo service busy.
1793	Incorrect PAD password.
1794	All the PAD virtual circuits are used
1795	X.25 initialization not possible.
179B	LCN not null in restart packet
179D	Incompatible header (receive error: all VC of concerned NS
179E	LCN greater than NBVC in NS directive
179F	Incorrect packet type
17A0	Invalid facility.
17B0	Normal disconnection.
17B1	X.25 Echo in use.
17B2	No more logical channels available.
17B3	No more PAD connections allowed.
17B4	TS SX25 or NU X25 object locked.
17B5	Buffer capacity overflow.
17B6	Normal disconnection.
17B8	Unknown calling SNPA (Sub-Network Point of Attachment).
17B9	Internet problem.
17CB	Call collision on VC
17CC	Incompatible generations (NR object without mapping).
17CE	Invalid status NR locked.
17CF	Lack of space.
17D0	Unknown subscriber.
17D4	TSCNX already used for another connection. SCF internal error.
17D7	Transport station locked.
17DD	Proper NS locked.
17DE	Invalid status NR locked.
17DF	Lack of space.
17E0	Forbidden parameter or invalid value.
17E1	Invalid transition.
17E2	Upward-mapped object (TS) not locked.

17E3	No object mapped above.
17E4	NR not locked (MP NR -ADD/-SUB) or virtual circuit already open.
17E5	NR is last in list and the TS is not locked.
17E6	No object mapped above (UP NR -PRIO). NR not mapped on TS.
17E7	Upward mapped object not locked
17E9	Mix of datagram and connection network
17EB	Class inconsistent with NR.
17EE	Incompatible generations. NR object without mapping.
17FF	Wrong parameter in administrative CALL
18xx	**Transport Layer**
1800	Normal disconnection initiated by the correspondent
1801	Local saturation at connection request time.
1802	Failed negotiation at connection time.
1803	Duplicate connection. Two or more requests have been issued for the same connection.
1804	Redundant request.
1805	Retransmission Time-out at transport level.
1806	Survey time-out at transport level.
1807	Transport protocol error.
1808	Session Control specified is not available (inaccessible).
1809	Requested Session Control Id unknown by remote transport.
180A	Termination because of disconnection by administration.
180B	Session Control/Transport interface error.
180C	Connection request on non-sharable VC in case of ISO Transport. ISO: header or parameter length is invalid.
1817	Station in shut-down state.
181F	No memory space at connection time.
1821	Session Control inaccessible by configured session routes. ISO: Session entity not attached to TSAP.
1824	Collision between Close NC and Open TC.
182E	Remote station not configured.
182F	Resource saturation.
1831	ISO: No route for the called NSAP.
1832	ISO: Received NSAP addresses are wrong.
1833	Segmentation violation.
1834	ISO:QOS priority not available temporarily, due to a local condition (for example, lack of resources).
1835	ISO:QOS priority permanently unavailable locally (for example, due to an error in the system generation).
183A	ISO: Remote reason not specified.
183C	ISO: Remote transport entity congestion at connect request time.
1840	Server in terminating state. TC has been re-assigned on another NC.

18A1	An additional NC has been assigned to a TC.
18B0	NC has been re-assigned on another VC.
18EF	Disconnection at Transport level caused by reception of RESTART DSA during the transfer phase.

Windows Sockets error Codes

Below is a list of Windows Sockets return codes and the corresponding description.

Hex code	**Windows Sockets Access Error name**	**Description**
2714	WSAEINTR	The (blocking) call was cancelled via WSACancelBlockingCall()
2719	WSAEBADF	The socket descriptor is not valid.
271E	WSAEFAULT	An invalid argument was supplied to the Windows Sockets API.
2726	WSAEINVAL	An invalid call was made to the Windows Sockets API.
2728	WSAEMFILE	No more file descriptors are available.
2733	WSAEWOULDBLOCK	The socket is marked as non-blocking and no connections are present to be accepted.
2734	WSAEINPROGRESS	A blocking Windows Sockets call is in progress.
2735	WSAEALREADY	The asynchronous routine being cancelled has already completed.
2736	WSAENOTSOCK	The descriptor is not a socket.
2737	WSAEDESTADDRREQ	A destination address is required.
2738	WSAEMSGSIZE	The datagram was too large to fit into the specified buffer and was truncated.
2739	WSAEPROTOTYPE	The specified protocol is the wrong type for this socket.
273A	WSAENOPROTOOPT	The option is unknown or unsupported.
273B	WSAEPROTONOSUPPORT	The specified protocol is not supported.

273C	WSAESOCKTNOSUPPORT	The specified socket type is not supported in this address family.
273D	WSAEOPNOTSUPP	The referenced socket is not a type that supports connection-oriented service.
273E	WSAEPFNOSUPPORT	
273F	WSAEAFNOSUPPORT	The specified address family is not supported by this protocol.
2740	WSAEADDRINUSE	The specified address is already in use.
2741	WSAEADDRNOTAVAIL	The specified address is not available from the local machine.
2742	WSAENETDOWN	The Windows Sockets implementation has detected that the network subsystem has failed.
2743	WSAENETUNREACH	The network address can't be reached from this host. There is probably a problem in the way you have set up TCP/IP routing for your PC (most likely you have not defined a default router).
2744	WSAENETRESET	The connection must be reset because the Windows Sockets implementation dropped it.
2745	WSAECONNABORTED	The connection has been closed.
2746	WSAECONNRESET	
2747	WSAENOBUFS	Not enough buffers available, or too many connections.
2748	WSAEISCONN	The socket is already connected.
2749	WSAENOTCONN	The socket is not connected.
274A	WSAESHUTDOWN	The socket has been shutdown.
274B	WSAETOOMANYREFS	
274C	WSAETIMEDOUT	Attempt to connect timed out without establishing a connection.
274D	WSAECONNREFUSED	The attempt to connect was forcefully rejected. The service on the other side is not available.
274E	WSAELOOP	Too many symbolic links were encountered in translating the path name.
274F	WSAENAMETOOLONG	
2750	WSAEHOSTDOWN	The host machine is out of service.
2751	WSAEHOSTUNREACH	The host machine is unreachable.

2752	WSAENOTEMPTY	
2753	WSAEPROCLIM	
2754	WSAEUSERS	
2755	WSAEDQUOT	
2756	WSAESTALE	
2757	WSAEREMOTE	
276B	WSASYSNOTREADY	Indicates that the underlying network subsystem is not ready for network communication.
276C	WSAVERNOTSUPPORTED	The version of Windows Sockets API support requested is not provided by this particular Windows Sockets implementation.
276D	WSANOTINITIALISED	A successful WSAStartup() must occur before using this API.
2AF9	WSAHOST_NOT_FOUND	Authoritative answer host not found.
2AFA	WSATRY_AGAIN	Non-authoritative answer host not found, or SERVERFAIL.
2AFB	WSANO_RECOVERY	Non-recoverable errors, FORMERR, REFUSED, NOTIMP.
2AFC	WSANO_DATA	Valid name, no data record of requested type.