HOST LINKS

G&R **GUFT**™

G&R Unified File Transfer

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

G&R GALLAGHER ROBERTSON

Version 6.3
© Gallagher & Robertson as 1990-2005
All Rights Reserved

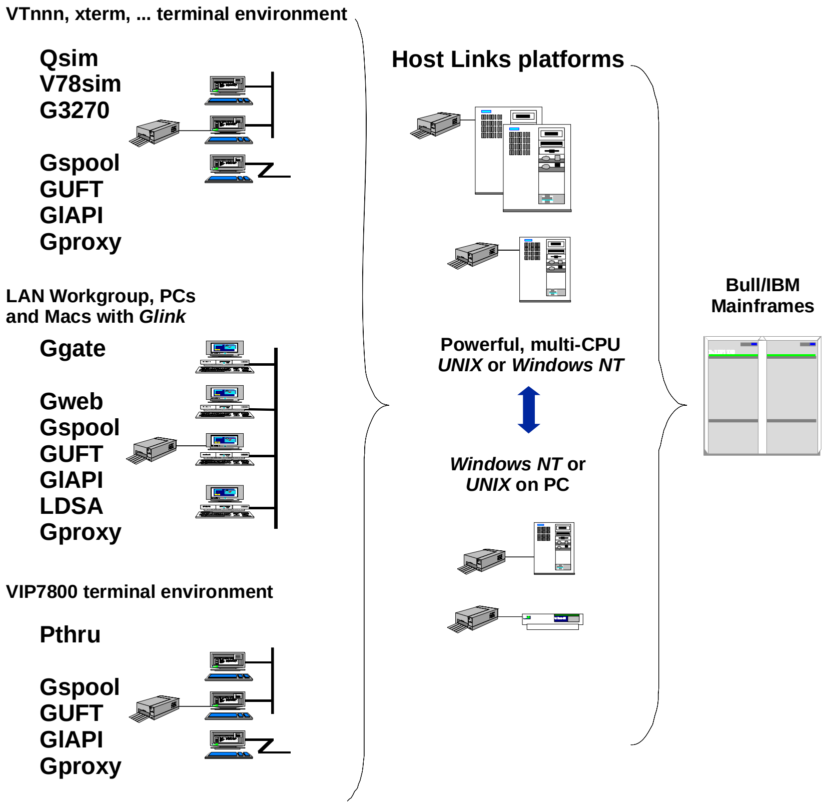
GALLAGHER & ROBERTSON AS, Kongens gate 23, N- 0153 Oslo, Norway
Tel: +47 23357800 • Fax: +47 23357801
www: http://www.gar.no/
e-mail: support@gar.no

Contents

Host Links GUFT

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.


VTnnn, xterm, ... terminal environment
Qsim
V78sim
G3270
Gspool
GUFT
GIAPI
Gproxy
LAN Workgroup, PCs and Macs with *Glink*
Ggate
Gweb
Gspool
GUFT
GIAPI
LDSA
Gproxy
VIP7800 terminal environment
Pthru
Gspool
GUFT
GIAPI
Gproxy
Host Links platforms
Powerful, multi-CPU *UNIX* or *Windows NT*
Windows NT or *UNIX* on PC
Bull/IBM Mainframes

Scope of the product

GUFT is an implementation of the Bull UFT file transfer protocols. These protocols enable transfer of data files between heterogeneous systems. The systems must be interconnected in a DSA/OSI network running over a private or public X25 network or over a local area network (LAN).

GUFT can be run on any UNIX/Linux or 32-bit Windows platform to which the *G&R Host Links* line handlers have been ported. This includes the Bull UNIX line as well as most other commercially available UNIX platforms and Windows servers. The GUFTw client can be run on any supported Windows platform.

There is an implementation of UFT on all the Bull minicomputers and mainframe systems.

UFT file transfers involve two DSA/OSI nodes. One of the nodes runs a UFT client, called a requester, and the other node runs a UFT server. The requester initiates the various file transfer requests and the server responds and performs the actions necessary to carry out the transfer.

The G&R UFT implementation includes both the requester and server protocol enabling file transfer between:

- A G&R/GUFT client and a G&R/GUFT server (called GUFTsrv)
- A G&R/GUFT client and a Bull UNIX, GCOS6, GCOS7, or GCOS8 UFT server.
- A Bull UNIX, GCOS6, GCOS7, or GCOS8 UFT client and a G&R/GUFT server.

Run-time licenses

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

License key	Description
`basic`	For the base G&R run-time system. Mandatory for all products.
`gline`	For the G&R line handlers
`guft`	For GUFT server, GUFTsrv (On UNIX/Linux the client is included)
`guftc`	For GUFT client, i.e. guft, guftw and the workstation redirector guftws

The licenses file identifies the G&R distributor, the owner of the license and the licensed products. The license key for a product will normally state how many users or simultaneous sessions the product is licensed for. If a limitation is specified in the license, only the licensed number of users or sessions can be active at any time.

UFT protocol services

The services that may be provided by a UFT client/server pair are as follows:

- basic communication services
    - connection to a remote system
    - disconnection from a remote system
    - interruption of a file transfer
- file management services
    - creation of a remote file
    - deletion of a remote file
    - request for information about a remote file
- file transfer services
    - transmission of a file to the remote system
    - reception of a file from the remote system
- extended services
    - restart of a file transfer

The mutual capabilities of a particular UFT client/server pair are negotiated after a session has been established between them.

Technical implementation

The UFT file transfer protocols are asymmetrical client/server rather than symmetrical peer to peer; i.e. the set of requests and responses available for exchange over a UFT session are different for the UFT requester and the server. In the G&R UFT implementation the requester and server are delivered as two separate programs.

The GUFT programs communicate with the remote system using the DSA/ISO protocols implemented in the *`Host Links`* line handlers (`gl_dsa`, `gl_diws`). The line handlers are documented in a separate manual (*`G&R/Gline`*).

GUFT, the UNIX/Linux client/requester

Screen mode

In full screen mode GUFT uses the *`G&R/Gvideo`* interface to present you with a user-friendly menu where the various UFT functions are offered. Additionally a lot of general features are offered, including the possibility of navigating around in the local file system, executing local commands and starting new UNIX/Linux shells.

You can have only a single session at a give time within a given context, but you may execute several contexts using the G&R (or other) context manager to start multiple simultaneous transfers. The UFT protocol allows transfer of only one file at a time, but you can do multiple file transfers serially and have several sessions with different UFT servers serially without terminating GUFT. All communications related parameters can be preconfigured, leaving you to simply select the local and remote file names and press the appropriate function key. While a file transfer is in progress GUFT updates the screen with status information (no. of bytes transferred, bytes per second and elapsed time).

Help is always available, and within the parameters menu the help is context sensitive to the parameter you are entering.

Command mode

In command mode you can initiate a single session to a single server. During this session you can send or receive a single file. GUFT command mode is suitable for unattended or background operation, and may be executed from a script.

In command mode a summary of the transfer is appended to a log and an accounting file `guft.log` and `guft.acc` in the directory where GUFT is started. See the section entitled *Log and accounting* in the *Administration* chapter, on page 56.

The Windows client/requester - GUFTw

Prerequisites

GUFTw runs on Windows 98, Windows NT 4.0, and Windows 2000. GUFTw normally communicates with the UFT server on the mainframe through the *G&R/Ggate* gateway, but can also use the *G&R/LDSA* communications stack running on the workstation.

Screen mode

In screen mode, GUFTw presents you with a user-friendly menu where the various UFT functions are offered. Additionally it offers an interface for navigating through the local file system and selecting files.

You can have a theoretically unlimited number of sessions at a time within a given instance of GUFTw. The UFT protocol allows transfer of only one file at a time, but you can do multiple connections to the same UFT server and have several transfers with different UFT servers running simultaneously. All communications related parameters could be pre-configured, leaving you to simply select the local and remote file names and press the appropriate toolbar button. While a file transfer is in progress GUFTw updates the screen with status information (no. of bytes transferred, bytes per second and elapsed time) and, when information on file sizes is available, a progress bar.

On-line help is available, and within the configuration dialog boxes the help is context sensitive to the parameter you are entering.

Command mode

Command mode for Windows workstations is available using a separate program, GUFTwX.EXE. In command mode you can initiate a single session to a single server. During this session you can send or receive a single file. GUFTw command mode is suitable for unattended or background operation, and may be executed from a script.

In command mode a summary of the transfer is appended to a log and an accounting file `guft.log` and `guft.acc` in the directory where GUFTwX is started. See the section entitled *Log and accounting* in the *Administration* chapter, on page 56.

The server - GUFTsrv

GUFTsrv runs in the background and 'listens' for connections to the DSA node. GUFTsrv accepts a single connection at a time but several GUFTsrv processes can be started listening for connects to the same mailbox name on the same DSA node. This requires use of the G&R DSA or DIWS listener. Please refer to the *`Gline`* documentation for details. GUFTsrv accepts file management and transfer requests issued by a remote UFT requestor. Files will be read from or written to the GUFTsrv local file system unless redirected to a client workstation by means of a special file name syntax in the requestor command. When a file transfer session is terminated, either normally by the requester at the end of the session, or forced by the server due to exceptions or protocol violations, GUFTsrv either terminates and leaves it to the listener to start up a new GUFTsrv instance when new connections come in, or, in the case where the listener is not used to start up GUFTsrv ‘on demand’, issues a new 'accept' command and waits for a new connection. In the latter case the server will run until you terminate it manually using the user/operator command or Gmanager.

File types supported

Local file types

The local UNIX/Linux file system does not have file types as such - the local file is always a sequential type i.e. the data is composed of data records that can only be accessed in their physical sequence. The ‘file type’ setting in GUFT mainly determines the size of the record (or ‘access unit’). The file’s access unit is also the transfer unit but the DSA session layer normally packs the records into transfer blocks of up to 16K Bytes (negotiated between the 2 systems) for more efficient transfer.

GUFT supports two types of sequential file access on the UNIX/Linux platforms:

- *VAR*

  The access unit is a logical record of variable length separated by the ASCII LF character (the LF itself is not transferred). Data code is normally set to *ASCII*, but code translation is possible if the remote system uses another code (e.g. *EBCDIC*). This type of file is sometimes referred to as a text file.

- *FIX*

  The access unit is a fixed, configurable size record. Data code will often need to be set to *BINARY* (no code translation will take place and no record separators will be inserted). This type of file is sometimes referred to as a byte-stream or a blocked file. This type of file access is useful when you want fast access and you do not want any logical record handling to take place.

Remote file types

GUFT supports access settings of 4 different types of remote files. The remote file type setting in GUFT is interpreted by the remote file system as follows:

- *VAR* - sequential text file, interpreted remotely as:

UNIX/Linux	same as for the local system
GCOS6	UFAS sequential file
GCOS7	UFAS sequential or source library files
GCOS8	GFRC ASCII (TSS format) file

- *FIX* - fixed record-size (blocked) file, interpreted remotely as:

UNIX/Linux	same as for the local system
GCOS6	UFAS sequential file with fixed record size
GCOS7	Binary library files
GCOS8	GFRC binary file with fixed record size

- *UFS* - sequential file, interpreted remotely as:

UNIX/Linux	UFAS is not applicable on remote UNIX
GCOS6	GCOS6 UFAS sequential (same as *VAR*)
GCOS7	GCOS7 UFAS sequential
GCOS8	GCOS8 UFAS sequential

- *UFR* - relative file, interpreted remotely as:

UNIX/Linux	UFAS is not applicable on remote UNIX
GCOS6	GCOS6 UFAS fixed relative (bound unit format)
GCOS7	Not Applicable
GCOS8	GCOS8 UFAS relative file

Exchange of other file types is not supported. In order to transfer unsupported types, the file must be converted to one of the supported types at the remote system before attempting the transfer.

Notes on problems related to usage of file types

If the user of the UFT requester gives the file type parameters incorrectly (e.g. in the GCOS7 EFTR command), the resulting file on the target system will normally not be accessible. For example, sending a text file in binary mode (i.e. as a *FIX* type) to a GCOS host implies that any logical record separators (e.g.

LFs) in the file will be transferred and the resulting GCOS file cannot be accessed as a sequential file. Likewise, if a GCOS file is sent in binary mode from the GCOS requester to the Host Links system, any attempt to access the file as a text file locally later (e.g. in a succeeding upload transfer request), will fail due to lack of record separators in the file.

In general transferring files using the *FIX* type (i.e. in ‘binary mode’) is often only meaningful between homogeneous systems.

In case of file transfers in *VAR* mode, the UFT server normally sets the maximum length of a logical record. In the GUFTsrv case, the default maximum is 512 bytes but this can be changed by the –SZ parameter (see parameter sections below). Some UFT implementations allow records larger than the negotiated maximum length to be received but will not allow sending such records. In *FIX* mode the negotiated record size is the (fixed) size of the access unit i.e. number of bytes read and transferred as a single unit (‘record’).

GUFT requester usage

Using the UNIX/Linux GUFT requester

GUFT can be used in two different modes: 'screen mode' and 'command mode'. The following documents the various parameters and commands used to control the execution of the program.

Parameters

If you start GUFT without parameters it runs in screen mode with default parameters picked up from the configuration file. GUFT gets its parameters from 3 different sources:

From the GUFT configuration file

The default configuration file is `guft.cfg` and its default location is under the standard G&R system directory:

UNIX/Linux	`/usr/gar/config/default/guft.cfg`

Additionally you may have your own configuration file located at:

UNIX/Linux	`/usr/gar/config/$LOGNAME/guft.cfg`

If a file with name `guft.cfg` is found in a directory with your $LOGNAME it will be used rather than the default configuration file.

From the command line

Command line parameters override configuration file parameters.

Interactively after you start GUFT

In screen mode you can set parameters at any time except during a file transfer.

The GUFT parameters

There are two types of parameters, GUFT parameters and line parameters.

The GUFT parameters are:

Parameter	**Description**
`-LF localfilename`	Name of the local file. The file name can be relative (i.e. just file name in the current directory) or absolute (i.e. include the file path).
`-RF remotefilename`	Name of the remote file. The file name can be relative or absolute (i.e. include the file path). The file path format is server dependent. If a relative name is used, the actual location of the file is server dependent. e.g. a remote file named myfile residing in a directory mydir belonging to user could have an absolute path name: GCOS6: `>udd>user>mydir>myfile` GCOS7: `my.sl..myfile` GCOS8: `user/mydir/myfile` UNIX/Linux: `/user/mydir/myfile` Windows: `c:\user\mydir\myfile`
`-CRF remotefilename`	Same as -RF (see above) except that the file does not exist on the server. GUFT will explicitly request the remote file to be created before the file transmission request (-put) is attempted. This parameter is applicable for 'command mode' only. Please observe that remote file creation is a negotiated facility.
`-GET`	Receive a file from the server using the given filenames (-LF, -RF). This parameter is applicable for 'command mode' only, and forces selection of command mode.

Parameter	Description
`-PUT`	Transmit a file to the server using the given filenames (-LF, -RF). This parameter is applicable for 'command mode' only, and forces selection of command mode.
`-LFT var/fix`	Local file type. You can choose to access the local file in 'var' or 'fix' mode. The default type is '“var”'. In ‘var’ mode, the file is divided in logical records of variable length separated by record separators (e.g. ASCII LF). In fixed mode the file is divided in fixed length access units. The size will default to 512 unless modified with the -SZ parameter.
`-RFT var/fix/ufs/ufr`	Remote file type. Sets the type of file on the remote system. Default type is var. Applicable file types depend upon the type of remote system. See the description of the –LFT parameter above and the section File types supported by GUFT on page 11.
`-SZ nnnn`	Record size in bytes. In ‘var’ mode the size is the maximum size of a logical record. In ‘fix’ mode the size is the fixed number of bytes read and transferred as a singe unit. The default size is 512 octets. The maximum size is server dependent.
`-DC ASC/EBC/BIN`	Data code used when transferring the file. Defaults to ASCII for all other UFT systems than DPS7, which uses EBCDIC. Can be set to BINary in order to provide transparent access (no translation, no record separator processing).
`-DBG`	Enables an internal trace.
`-NL`	No logging or accounting. In command mode a short log of GUFT activity is appended to the GUFT log and to the account file. The GUFT log is located in the directory where you start GUFT and is named guft.log. You use this parameter to suppress logging and accounting.

Parameter	Description
`-RS (crlf)`	Force insertion of record separators also for FIX file types. If no parameter value is given, the record separator defaults to LF. CRLF can be given as parameter value in which case both a CR and a LF will be used to separate records. This might be desirable for NFS mounted file systems that are accessed by programs that need PC style delimiters. When transmitting, the record separators are not sent. Also, in the CRLF case, a record containing a single CNTL Z (hex 1A) character will be interpreted as an end of file marker (the end of file marker itself is not transmitted). Note; this parameter should not be needed, and indicates some sort of protocol incompatibility between the server and the requester.
`-APN`	Open the local file in 'append mode' i.e. add new records at the end of the existing file.
`-CN`	Auto-connect at start-up using the default parameters (applicable to screen mode).
`-XL xx`	Transliterate between 7-bit national characters on the remote system and the ISO/DO11 8 bit equivalents on the local system. The correct `-XL` (`GB, GE, FR, SF, DE, NO, SP, IT, JA`) must be specified if you communicate with a 7-bit national host.
`-NC`	Force no compression in the transmission request to the UFT server.
`-ATR on/off`	Enable/disable file size attribute support. Defaults to off.

Line parameters

You can set all line parameters in the command line. The line parameters must follow the GUFT parameters and must be preceded by the `-LI` parameter. The line parameters are described in the *`G&R/Gline`* documentation. A brief explanation of the most relevant parameters follows:

Parameter	Description
`-LI dsa/diws`	Select native DSA or DIWS protocol (mandatory if more line parameters are given).
`-DN nodename`	DSA/ISO default 'node' name of the remote system. This node must be configured in the `dsa.cfg` configuration file.
`-HM hostmode`	Specify the type of remote system (DPS8, DPS7, DPS6 or UNIX) to which you connect. Defaults to DPS8.
`-DU username`	The name of the local (GUFT) user. In order to ensure correct access rights on the remote file system this user name must in most cases be known to the remote system. For instance, in order to access GCOS8 a certain catalog structure must be set up using this name (refer to the Server specific information section on page 58). There is no default.
`-DA applicationname`	Name of the UFT mailbox on the remote system. Defaults to FILETRAN. The mailbox name of GUFT will default to 'GUFT' but can be changed by the `-mn` parameter.
`-PW password`	Specifies the password of the local user on the remote system. Used mainly by GCOS7 servers
`-DB billing`	Specifies the 'billing' used on the remote system. Used mainly by GCOS7 servers.
`-DP project`	Specifies the project name used on the remote system. Used mainly by GCOS7 servers.
`-S_`	Enables 'session' interface trace, with details of line module events and data.

Note that all outgoing connects use the default local node name to identify the calling node to the server. The calling node name may be used for security in the file access control logic on the server. If you also use GUFTsrv to accept connects from other UFT systems, then you may want to reserve the default local node name for GUFTsrv usage in order to minimize configuration of access control files on the remote systems.

GUFT start-up examples

Given the following `guft.cfg` file:

```
Default
  -li dsa
  -hm dps8
  -dn b6dd
  -du jim
```

Start GUFT in screen mode with GUFT debug enabled. You will set file names etc. later interactively:

```
   guft -DBG
```

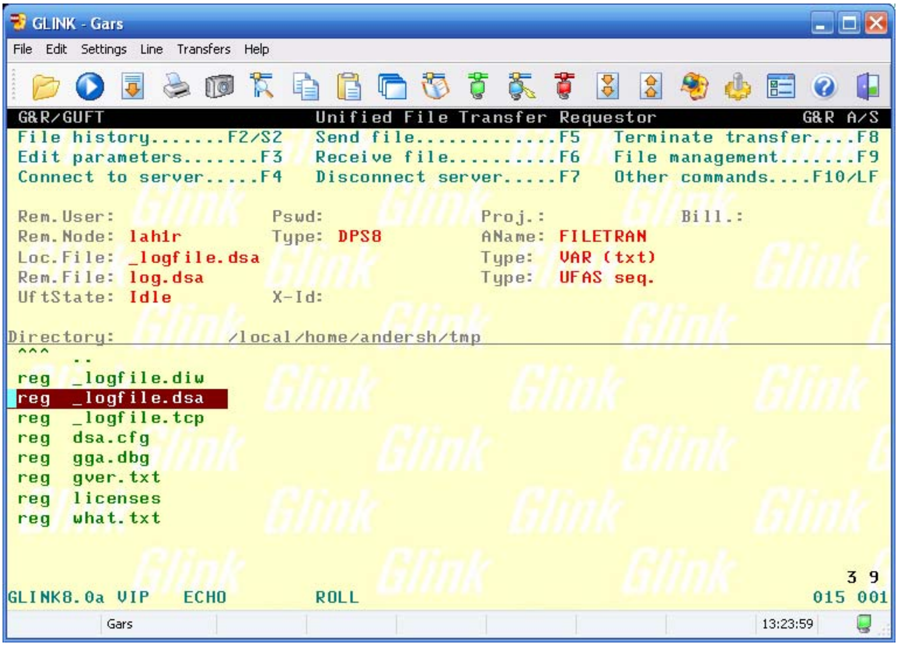
Start GUFT in command mode, and send a local (defaults to *VAR* type of) file to the remote system where the file does not yet exist:

```
   guft -lf myfile -crf >udd>jim>myfile -put
```

Start GUFT in screen mode overriding the file mode and size. Give the node name and enable full 'session trace':

```
guft -lft fix -rft fix -dc bin -sz 256 -li dsa -dn en3c
     -s_
```

Running GUFT in screen mode



```
G&R/GUFT                    Unified File Transfer Requestor                  G&R A/S
File history.......F2/S2    Send file.............F5    Terminate transfer....F8
Edit parameters.......F3    Receive file..........F6    File management.......F9
Connect to server.....F4    Disconnect server.....F7    Other commands....F10/LF

Rem.User:              Pswd:              Proj.:                Bill.:
Rem.Node:  lahir       Type:  DPS8        AName:  FILETRAN
Loc.File:  _logfile.dsa                   Type:   VAR (txt)
Rem.File:  log.dsa                        Type:   UFAS seq.
UftState:  Idle        X-Id:

Directory:          /local/home/andersh/tmp
 ^^^    ..
 reg    _logfile.diw
 reg    _logfile.dsa
 reg    _logfile.tcp
 reg    dsa.cfg
 reg    gga.dbg
 reg    gver.txt
 reg    licenses
 reg    what.txt

                                                                             3  9
GLINK8.0a VIP   ECHO         ROLL                                          015 001
```

If you start GUFT without a `-get` or `-put` parameter on the command line, it starts in screen mode. You can terminate GUFT at any time by pressing `ESC` twice or by pressing `LF` followed by `Q`.

GUFT starts by analyzing the parameters on the `guft.cfg` configuration file (if any) and from the command line. It then displays a screen that is divided into 3 sections:

- A 'menu section' that documents the command interface. You use function keys to trigger most commands.
- A 'parameter section' showing the current parameter settings.
- A 'file system section' interface similar to the G&R file navigation facility (*G&R/Gdir*).

The GUFT screen interface

F1 - Help

Help is available at any time. The help is implemented using the standard *`G&R/Gmenu`* subsystem enabling you to modify or expand the content of the help screens 'on the fly'. The help documents the command interface. If you press F1 while editing GUFT parameters the help is context sensitive to the field you are entering. You navigate in the help system using the cursor keys. You return to the main GUFT screen by pressing the 'ENTER' key.

F3 - Edit parameters

Enables you to dynamically change the UFT file and line parameters during the UFT session.

The 1st line of parameters holds user/identification parameters.

The 2nd line holds the server name, type and UFT application name.

The 3rd line holds information about the local file.

The 4th line holds information about the remote file.

The parameters are initialized with information from the configuration-file and/or command line. You can edit some of the fields. Others have preselected values (use cursor left/right to choose). You can use cursor up/down and tab/back-tab to navigate between the fields. Function keys or `ESC` take you out of edit mode. The field information is validated and if it is inconsistent you will be positioned to the field for reentry. You may call context sensitive help by pressing `F1` when positioned in a field.

Refer to the *GUFT parameter* section for a detailed description of the various fields.

Field	**Description**
`LocUser`	The name of the local user *`Gline`* parameter = `-DU xxxxxx`
`Pswd`	The password of the local user *`Gline`* parameter = `-PW xxxx or -D? xxxx`
`Proj`	Project, used for GCOS7 connections only *`Gline`* parameter = `-DP xxxx`
`Bill`	Billing, used for GCOS7 connections only *`Gline`* parameter = `-DB xxxx`
`RemNode`	Specify name of the remote node *`Gline`* parameter = `-DN xxxx`
`Type`	Host type *`Gline`* parameter = `-HM DPS6/DPS7/DPS8/UNIX`
`AName`	The remote UFT application name (mailbox) *`Gline`* parameter = `-DA xxxxxxx`
`LocFile`	The local file name. Relative or absolute path name. The 16 most recent filenames is remembered and can be selected from a file name menu which is enabled by the `F2` key while positioned in the parameter menus.
`Type`	Type of the local file *VAR* (text type) or *FIX* (binary type)
`Remfile`	The remote file name. name. Relative or absolute path name. The 16 most recent filenames is remembered and can be selected from a file name menu which is enabled by the `SF2` key while positioned in the parameter menus.
`Type`	Type of the remote file Valid selections: *VAR*, *FIX*, *UFAS Seq* or *UFAS Rel* Defaults to *VAR*

F4 - Connect to server

Initiates a connection to the target UFT server using the current line parameter settings. For the connection to be successful the underlying communications software must be correctly configured.

The result of the connection attempt is reported on the status line. A successful connection results in a UFT application dialog that in turn will generate a transfer id (Xfer Id). At this point the user can select, create or delete remote files and initiate transfers.

F5 - Send file

Initiates a send file operation using the local and remote files that are selected. The progress of the file transfer is indicated in the parameter area of the screen. You may terminate the transfer at any time with the F8 key. An interrupted transfer will leave the files in an undefined state. A successful transfer is indicated by a 'File successfully sent' message on the status line. If the session is not already opened, a connect request will be launched automatically.

F6 - Receive file

Initiates a receive file operation using the local and remote files that are selected. The progress of the file transfer is indicated in the parameter area of the screen. You may terminate the transfer at any time with the F8 key. An interrupted transfer will leave the files in an undefined state. A successful transfer is indicated by a 'File successfully received' message on the status line. If the session is not already opened, a connect request will be launched automatically.

F7 - Disconnect from server

Issues an orderly session disconnect command to the remote system. It will not work during the 'Transfer' state. You must issue a disconnect request before you attempt a new connect to a different (or the same) server. A successful disconnection results in a 'Disconnected by user/application' message on the status line.

F8 - Terminate transfer

You can interrupt a file transfer with the F8 key. Once a transfer is started the sender normally continues to send data until the complete file is transferred. The transfer can only be interrupted by sending a special 'expedited' message to the sender against the flow of data. This will only work if the underlying transport provider stack supports expedited data (i.e. the result is platform dependant).

F9 - File management

This is a selection of UFT file management functions that can be performed locally or on the UFT server. If a session is established when F2 is pressed, you will be prompted for local or remote management. Otherwise only local functions are available.

File Creation

File creation on the local system is not necessary since local files that do not exist, will be created automatically. The remote file must be explicitly created on some servers. An initial file size must be given. The result of the creation request is reported on the status line.

File deletion

For local deletion, the file you are positioned on will be suggested. You will be asked to confirm. For remote deletion the current selected file, if any, will be suggested. You will not be asked to confirm the delete. The result of the deletion request is reported on the status line.

Attributes

You can request various attributes of a remote file. The availability of this service is negotiated while establishing the session.

LF - Other commands

This is a selection of other, less frequently used commands activated by pressing the command key followed by a single character:

E	Execute a local (UNIX/Linux) command. After the command is executed you will be asked to press ENTER to return to GUFT.
!	Starts a UNIX/Linux shell. The 'exit' command takes you back to GUFT.
B	Set transfer block size (= max. record size). Max. = 8192, default = 512
X	Set transfer mode to *Line* (applicable to *VAR* file type) or *Block* (for *FIX* file type)
C	Toggles data compression. Data compression can speed up the data transfer. The effect of the compression depends on the type of file and the file content.
D	Toggles Debug/trace on and off.
O	Toggles open mode between *new* (default) and *append*.
A	Toggles attribute support.
D	Sets data code to *ASCII*, *EBCDIC* or *BINARY*
Q	Quits/terminates the program.

File transfer status

When a file transfer starts, GUFT updates the screen with status information. The 'State line' contains information about the current state of the UFT session. The following states will be indicated:

Idle	Not connected to server
Session	A session with an UFT server is established. A transfer ID is generated
File	Files are selected
Open	Files are opened (a transit state between *File* and *Transfer*)
Transfer	A file transfer is in progress

In transfer state GUFT shows the amount of data transmitted, the transfer rate (characters pr. second) and the elapsed time. While sending the number of transmitted characters shows what has been delivered to the underlying communication layers, the data is not necessarily transmitted on the line. Generally this will show an apparently high speed at first until the underlying communications layers have filled all their buffers, and thereafter will steady down to the true rate of transfer over the line. While receiving it shows the actual number of characters received into GUFT. If compression is enabled the count is performed after decompression.

If file size attribute support is enabled by means off the `-atr` parameter or via `LF A`, then the size of the target file will be indicated if given by the server. When sending a file the size will be the same as the size of the 'source' file. When receiving a file the server will ask for the size of the file in the servers file system and report this to the requester. Please observe that this size is not always the exact number of bytes in the file in that it can include file system overhead.

Running GUFT in command mode

You can start GUFT from the command line and run without the screen interface i.e. unattended from a script. In command mode GUFT takes parameters from the GUFT configuration file and from the command line. If you use the `-get` or `-put` parameter in the command line GUFT starts in command mode, issues the `-get` or `-put` request and terminates after the transfer. You must also specify the local and remote file names (`-LF -RF`). A special format of the `-RF` command `-CRF` allows you to create the remote file before a file transmission (`-put`) request is issued.

In command mode a summary of the transfer is appended to a log and an accounting file `guft.log` and `guft.acc` in the directory where GUFT is started. See the section entitled *Log and accounting* in the *Administration* chapter, on page 56.

Using the Windows GUFT requester

The Windows GUFT requester, called GUFTw, can be used in two different modes: 'screen mode' and 'command mode'. The following documents the various parameters and commands used to control the execution of this program.

Parameters

GUFTw gets its parameters from 2 different sources:

From the GUFTw configuration file

The default configuration file is `guft.ini` and its default location is under the standard G&R configuration directory:

```
Windows   \gar\config\default\guft.ini
```

If this configuration file exists, it is only used on the first start-up of GUFTw to load a default configuration. Thereafter your own configuration file is created in a directory with your Windows login identity, and the first GUFTw configuration you use is saved as the default configuration:

```
Windows   \gar\config\LOGIN\guft.ini
```

The G&R configuration directory may be shared with others if it is placed on the file server, or the installation process may build it locally on your workstation.

Interactively after you start GUFTw

In screen mode you can set parameters at any time.

GUFTw start-up examples

Given the following `guft.ini` file:

```
[GUFT]
DefaultHostParams=-li dsa -hm dps6 -dn b7dd -du jim
```

Start GUFTw by double-clicking its icon in the Gallagher & Robertson program group. You will set file names etc. later interactively.

Start GUFTw in command mode, and send a local *VAR* file to the remote system where the file does not yet exist:

```
guftwx.exe -lf myfile -crf >udd>jim>myfile -put
```

Running GUFTw in screen mode

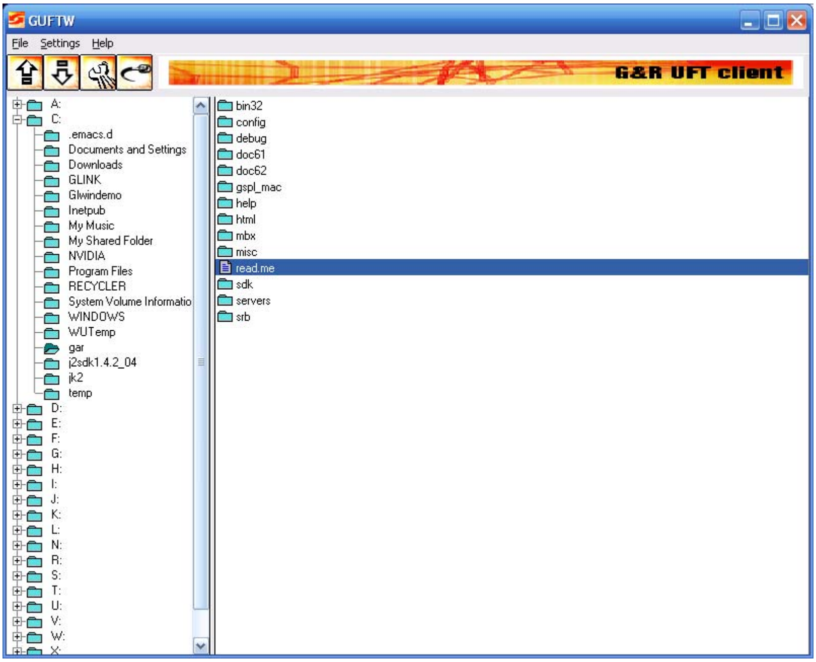
When you start GUFTw, it starts in screen (Windows GUI) mode. You can terminate GUFTw at any time by selecting Exit from the File menu, clicking the Exit button in the toolbar, or pressing Alt-F4.

GUFTw starts by analyzing the parameters on the `guft.ini` configuration file (if any) and from the command line. It then displays a window divided into three sections:

- A menu bar with menu options for each GUFT command.
- A toolbar with buttons for the most commonly used commands.
- A file system interface similar to the Windows Explorer.

The GUFTw screen interface

Sample of the main GUFTw window:

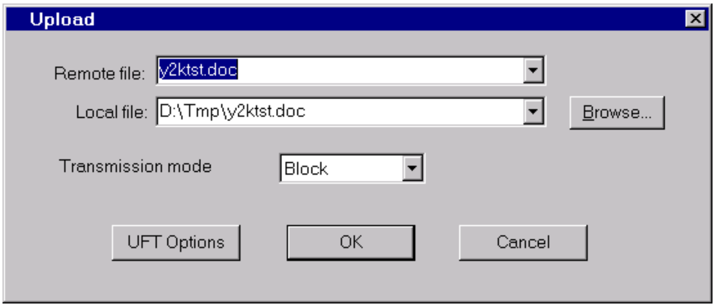


The main GUFTw window presents the local file system using a typical Windows file tree format. This allows the user to navigate and select files using the mouse and/or the keyboard. The following options are available from the menu bar and the tool bar:

File

Menu entries for upload and download. These functions are also on the toolbar.

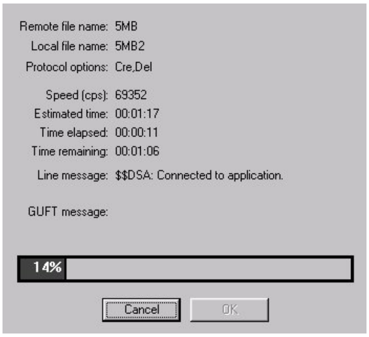
Upload



Initiates a send file operation. You will first be prompted for a local and remote file name, by default set to the file currently selected in the file navigation window. The 16 most recently used filenames are available from the drop down menu. You can navigate in the local filesystem using the 'browse' option. You can go directly to the 'UFT option' window if the current transmission mode is not correctly set. When the 'OK' button is pressed, a new window will appear showing the progress of the file transfer. You may terminate the transfer at any time with the Esc key, or by clicking 'Cancel.' An interrupted transfer will leave the files in an undefined state.

Download

Initiates a receive file operation. It uses the same window and options as described for 'Upload' above. In this case, the size of the file to be received will only be available in the transfer windows if 'file attributes' were negotiated with the UFT server.

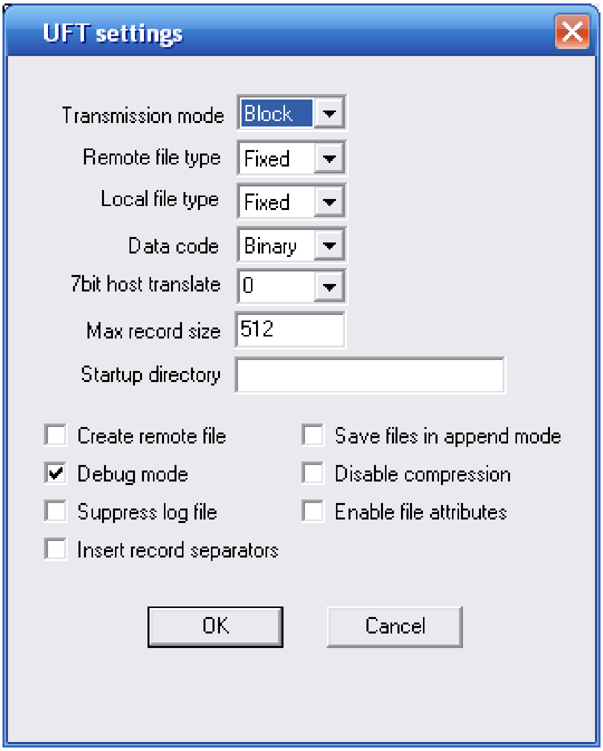


Remote file name: 5MB
Local file name: 5MB2
Protocol options: Cre,Del
Speed (cps): 69352
Estimated time: 00:01:17
Time elapsed: 00:00:11
Time remaining: 00:01:06
Line message: $$DSA: Connected to application.
GUFT message:

Settings

UFT settings

The UFT settings window:


UFT settings

Transmission mode: Block
Remote file type: Fixed
Local file type: Fixed
Data code: Binary
7bit host translate: 0
Max record size: 512
Startup directory:

- [ ] Create remote file
- [✓] Debug mode
- [ ] Suppress log file
- [ ] Insert record separators
- [ ] Save files in append mode
- [ ] Disable compression
- [ ] Enable file attributes

Enables you to change the UFT transfer parameters. When you select this command, or click the corresponding button on the toolbar, you will be presented with a dialog box where you can change the transfer parameters. The parameters are initialized with information from the configuration file. When you click 'OK', the field information is validated and if it is inconsistent you will be positioned to the field for reentry. You may call context sensitive help by pressing F1 when positioned in a field.

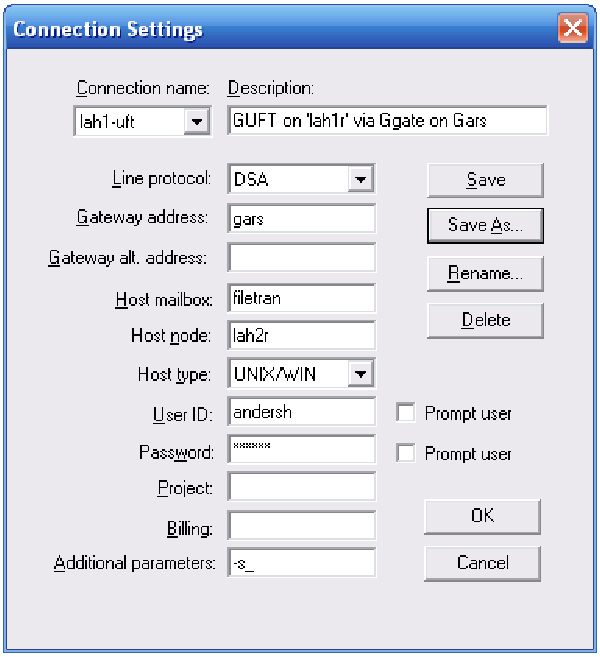
The available parameters are:

Field	**Description**
Transmission mode	'Line' or 'Block'. When set to 'line', automatically sets filetypes to 'var' and datacode to 'ascii' or 'ebcdic' (depending on host type). When set to 'block', automatically sets file types to 'fix' and datacode to 'binary'. The automatic setting of the other fields can be manually overridden.
Local file type	Default local file type. You can choose to access the local file in variable or fixed size mode. The default type is *VAR*. In *VAR* mode, the file is divided in logical records of variable length separated by record separators (e.g. ASCII LFs). In *FIX* mode the file is divided in fixed length records. The access unit will default to 512 bytes unless modified in the "Record size" field parameter. Command mode parameter: `-LFT VAR/FIX`
Remote file type	Remote file type. Sets the type of file on the remote system. Default type is *VAR*. Applicable file types depend upon the type of remote system. See the description of the `-LT` parameter above and the section File types suppor-ted by GUFT on page 11. Command mode parameter: `-RFT VAR/FIX/UFS/UFR`
Record size	Record size in bytes. In 'var' mode the size is the maximum size of a logical record. In 'fix' mode the size is the fixed number of bytes read and transferred as a singe unit. The default size is 512 octets. The maximum size is server dependent.
Create remote file	GUFTw will explicitly create the file before the file transmission request is attempted. Please observe that remote file creation is a negotiated facility. Command mode parameter: `-CRF`
Debug mode	Enables an internal trace of GUFT events. Command mode parameter: `-DBG`

Field	Description
Suppress log file	Normally, a short log of GUFT activity is appended to the GUFT log. The GUFT log is located in the directory where you start GUFTw and is named `guft.log`. Command mode parameter: `-NL`
Insert record separators	Force insertion of record separators (LF) in the local file (i.e. also in the case of a 'fix' type of transmission). Command mode parameter: `-RS`
Save files in append mode	When receiving, opens the local file in 'append mode' (adds new records at the end of the existing file). Command mode parameter: `-APN`
Use ISO/DS11 transliteration	Transliterate between national 7-bit characters on the remote system and the ISO/DO11 8 bit equivalents on the Windows system. Command mode parameter: `-XL`
Disable compression	Force no compression in the transmission request to the UFT server. Command mode parameter: `-NC`
Enable file attributes	By default, GUFTw does not attempt to negotiate file attribute information with the host UFT server and GUFTw will not be able to display detailed status information during downloads. When the 'Enable file attributes' entry in the 'UFT settings' menu is checked, GUFTw will attempt to negotiate file attribute information with the host UFT server. If this is supported by the UFT server, GUFTw will be able to obtain the file size from the host and display a progress bar which shows how much of the file has been transferred, and also estimate the total duration of the transfer as shown here:

Line settings

The line settings window:


Connection Settings

Connection name: lah1-uft
Description: GUFT on 'lah1r' via Ggate on Gars
Line protocol: DSA
Gateway address: gars
Gateway alt. address:
Host mailbox: filetran
Host node: lah2r
Host type: UNIX/WIN
User ID: andersh
Password: ******
Project:
Billing:
Additional parameters: -s_

Enables the user to change the UFT line parameters. When this command is selected, or the corresponding button on the toolbar is clicked, the user will be presented with a dialog box where the communication line parameters can be changed. The parameters are initialized with information from the configuration file. When 'OK' is clicked, the field information is validated and if it is inconsistent, the user will be positioned to the field for reentry. The user may call context sensitive help by pressing `F1` when positioned in a field.

The available parameters are:

Field	**Description**
Connection name	Allows you to load a previously defined (in the Host Links system, 'dsa.cfg' file) set of line parameters.

Field	**Description**
Description	This field is not actually used by GUFTw, but can help you remember differences between saved connection names.
Line protocol	Allows you to select native DSA or DIWS protocol. `Gline` parameter: `-LI xxxx`
Gateway address	This field must be entered. It contains the IP address or host name of a server running Ggate. GUFTw uses Ggate to establish a DSA connection with the UFT server. `Gline` parameter: `-LI DSA/DIWS:xxxx`
Host application mailbox	Name of the UFT application/mailbox on the remote system. Defaults to FILETRAN. `Gline` parameter: `-DA xxxx`
Host node	DSA/ISO default 'node' name of the remote system. It must be known in the Ggate system `Gline` parameter: `-DN xxxx`
Host type	Specify type of remote system (DPS8, DPS7, DPS6, UNIX) to which you want to connect. Defaults to DPS8. `Gline` parameter: `-HM xxxx`
User ID	The name of the local (GUFTw) user. In order to ensure correct access rights on the remote file system this user name must in most cases be known to the remote system. For instance, in order to access GCOS8 a certain catalog structure must be set up using this name (refer to the *Server specific information* section on page 58). No defaults. `Gline` parameter: `-DU xxxx`
Password	Specifies the password of the local user on the remote system. Used mainly by GCOS7 servers. `Gline` parameter: `-D? xxxx`
Project	Specifies the project name used on the remote system. Used mainly by GCOS7 hosts. `Gline` parameter: `-DP xxxx`

Field	Description
Billing	Specifies the 'billing' used on the remote system. Used mainly by GCOS7 servers. `Gline` parameter: `-DB xxxx`
Additional parameters	Any parameters entered in this field will be sent to Ggate after all the others. A possible option here is `-s_`. This will enable a 'session' interface trace. Session events and line data will be written to a 'debug' file on the Ggate system.
Save	Saves the currently entered set of line parameters to the `guft.ini` file under the name selected in the "Con-nection name" drop-down box. If the "Connect name" box is empty, you will be prompted for a new name.
Save As	Saves the current set of line parameters to the `guft.ini` file under a new name. You will be prompted for a con-nection name.
Delete	Deletes the parameter set selected in the "Connection name" drop-down box from the `guft.ini` file.
Rename	Renames the set of parameters currently selected in the "Connection name" drop-down box.

Exit

Terminates GUFTw.

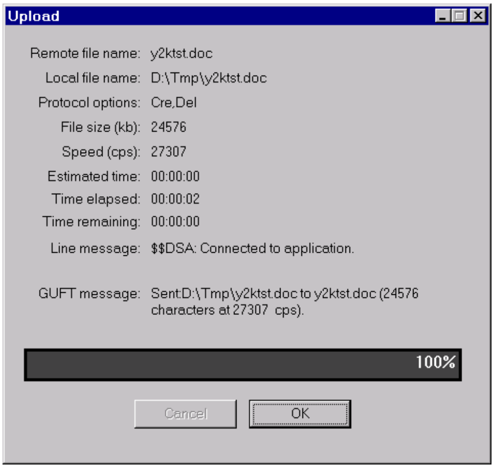
Help

Help is available at any time. The help documents the command interface. If you press F1 while editing GUFTw parameters the help is context sensitive to the field you are entering.

Transfer status

When a file transfer starts, GUFTw creates a new window to show the progress of the transfer. This window shows the amount of data transmitted, the transfer rate (characters pr. second) and the elapsed time. If information on file sizes is available, it will also display a “progress bar” showing the percentage of the transfer which has been completed, and time estimates for how long the transfer will take to complete.

While sending the number of transmitted characters shows what has been delivered to the underlying communication layers; the data is not necessarily transmitted on the line. Generally this will show an apparently high speed at first until the underlying communications layers have filled all their buffers, and thereafter will steady down to the true rate of transfer over the line. While receiving it shows the actual number of characters received. If compression is enabled the count is performed after decompression.



Remote file name:	y2ktst.doc
Local file name:	D:\Tmp\y2ktst.doc
Protocol options:	Cre,Del
File size (kb):	24576
Speed (cps):	27307
Estimated time:	00:00:00
Time elapsed:	00:00:02
Time remaining:	00:00:00
Line message:	$$DSA: Connected to application.
GUFT message:	Sent D:\Tmp\y2ktst.doc to y2ktst.doc (24576 characters at 27307 cps).

Running GUFTw in command mode

You can start GUFTw from the command line and run without the screen interface i.e. unattended from a script. To run GUFTw in command mode, you must launch it as GUFTwX, instead of GUFTw. GUFTwX takes parameters from the GUFT configuration file and from the command line. You must use the `-get` or `-put` parameter in the command line to start GUFTwX in command mode. GUFTwX issues the `-get` or `-put` request and terminates after the transfer. You must also specify the local and remote file names (`-LF -RF`). A special format of the `-RF` command `-CRF` allows you to create the remote file before a file transmission (`-put`) request is issued.

If you are running GUFTwX in command mode from an unattended script the default is to not display dialog boxes if an error occurs. If you will be present and would like to see and OK any error messages you must set the `-v` parameter.

For more information on other available parameters, see the section on GUFTw screen interface on page 31.

In command mode a summary of the transfer is appended to a log and an accounting file `guft.log` and `guft.acc` in the directory where GUFTwX is started. See the section entitled *Log and accounting* in the *Administration* chapter, on page 56.

GUFT requester filename history

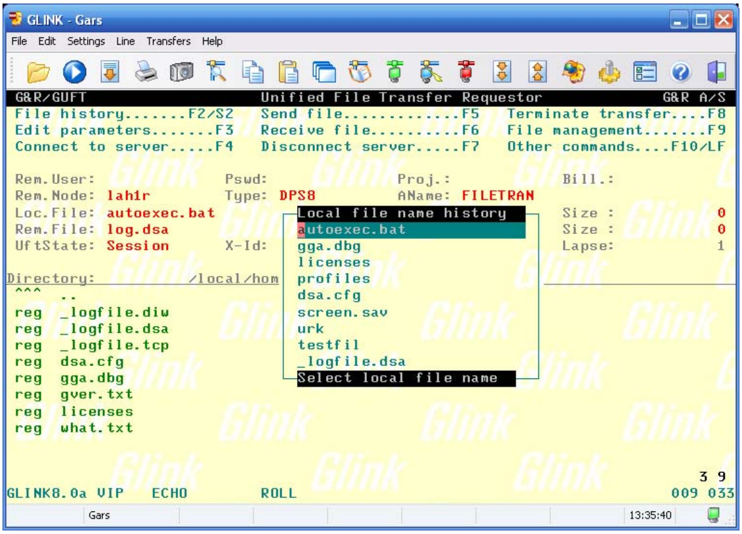
The GUFT requesters now remember the 16 most recently used local and remote filenames that has been transferred. Every time a file transfer is successfully executed, a filename history file is updated in the user configuration structure. The file is called:

Windows	`\gar\config\USERNAME\guft.hst`
UNIX/Linux	`/gar/config/USERNAME/guft.hst`

and can, if necessary be manually edited. The file contains 32 logical records (16 local file entries and 16 remote file entries) and the format of the records is:

File type, 1 character
    0 = text file, 1 = fixed file, 2 = UFAS sequential, 3= UFAS relative
File name, variable size (max 64) characters

The following shows a GUFT requester window with the filename history menu activated:



```
G&R/GUFT                    Unified File Transfer Requestor              G&R A/S
File history.......F2/S2   Send file.............F5   Terminate transfer....F8
Edit parameters.......F3   Receive file..........F6   File management.......F9
Connect to server.....F4   Disconnect server.....F7   Other commands....F10/LF

Rem.User:                Pswd:               Proj.:                Bill.:
Rem.Node: lahir          Type: DPS8          AName: FILETRAN
Loc.File: autoexec.bat         Local file name history            Size :       0
Rem.File: log.dsa              autoexec.bat                       Size :       0
UftState: Session        X-Id: gga.dbg                            Lapse:       1
                               licenses
Directory:         /local/hom  profiles
 ^^^    ..                     dsa.cfg
 reg  _logfile.diw             screen.sav
 reg  _logfile.dsa             urk
 reg  _logfile.tcp             testfil
 reg  dsa.cfg                  _logfile.dsa
 reg  gga.dbg                  Select local file name
 reg  gver.txt
 reg  licenses
 reg  what.txt

                                                                             3 9
GLINK8.0a VIP   ECHO        ROLL                                         009 033
```

GUFT server usage

Starting GUFTsrv

GUFTsrv accepts incoming sessions from a UFT requester. In the Windows case and also on UNIX/Linux if the RFC1006 protocol (i.e. DSA session over TCP/IP transport) is to be used, you **must** start the DSA listener first. Even though the listener is not an absolute requirement in a DSA configuration on UNIX/Linux, use of the listener is generally recommended because it allows several UFT sessions to run simultaneously.

On Demand: On either platform the DSA listener can be configured to start GUFTsrv on demand when a connect request is received for a local UFT mailbox. See the section on Starting GUFTsrv on demand on page 45 for more information. This is the recommended way of starting GUFTsrv. Note that 'on demand' startup cannot be used if the GUFTsrv process is executing on some remote Host Links system (i.e. accessed via Ggate).

UNIX/Linux: You can start GUFTsrv at boot time by including the GUFTsrv command line in the `/etc/inittab` file. For more information see the *Installation & Configuration* manual for *Host Links on UNIX/Linux*. GUFTsrv can also be started as an operator command, usually for testing.

Windows: You can start GUFTsrv as a Windows service by including the GUFTsrv startup command in the *`Gservice`* configuration file. For more information on *`Gservice`*, see the *Installation & Configuration* manual for *Host Links on Windows*. GUFTsrv can also be started using Gmanager to enter a command line that will be passed to Gservice for launching, usually for testing.

Accepting incoming connections

The command lines that start copies of GUFTsrv should be configured in the listener’s configuration file. The DSA listener will then start new instances of GUFTsrv when it receives connect requests for the mailbox name specified in the command line. See the section on Starting GUFTsrv on demand on page 45.

If start on demand is not used, GUFTsrv must be started manually before a remote UFT requester command is launched.

The node name of the Host Links platform must be configured as a remote node in the remote system (i.e. Bull Mainframe, Datanet, MainWay, GCOS6 or another UNIX/Linux system) which will be issuing the connect request. The mailbox name to which the application connects must be the one configured for GUFTsrv (defaults to FILETRAN, can be changed by –MN parameter)

A 'connect accept' is registered with the listener if GUFTsrv is configured for start on demand. The DSA listener forwards the incoming connection to GUFTsrv when it arrives.

Multiple incoming connections

If multiple copies of GUFTsrv for UNIX are to be run simultaneously, or if GUFTsrv is to coexist with other *Host Links* products that accept connect requests, you must remember to start the *Host Links* listener program before you start any of the GUFTsrv copies. On Windows, and on UNIX/Linux if RFC1006 is to be used, the listener **must** always be running, even if only one GUFTsrv is to be started. The use of the listener is recommended in all cases.

The *Host Links* listener supports multiple incoming connections to the same node and even the same mailbox. This daemon listener program must be started before the programs accepting incoming connections. When the connect requests arrive for the different mailboxes, the listener program forwards the connections to the programs waiting for them, or starts the program configured for the mailbox name in the listeners configuration file.

See the *Gline* manual for more information about the listener.

Start on demand

A configuration file controls the mapping from the DSA mailbox name (`-mn`) and, optionally, extension (`-mx`) to the command line for the GUFTsrv to be started. There is one file for each DSA node name (`SCID`) for which connections are being accepted. It is placed in the `servers` directory:

Windows	`\gar\servers\<scid>.gli`
UNIX/Linux	`/usr/gar/servers/<scid>.gli`

The file is either `config.dsa` or `config.diw` depending on whether the listener (`nl_dsa`) uses native DSA (`-prot dsa`) or DIWS (`-prot diws`).

Example file `/usr/gar/servers/grdl.gli/config.dsa`:

```
* UFT server
listen -mn filetran -cmd guftsrv
```

When a connect request arrives for mailbox name `filetran` on node `grdl`, the configured GUFTsrv is started. The GUFTsrv command does not need to include the `–LN` parameter or the `-ID` parameter, it is implicit. When GUFTsrv is started this way, it will terminate after each complete File transfer session

Note that multiple simultaneous connects to the mailboxes are allowed, and multiple instances of the GUFTsrv will be started unless this is disabled with the `-LIM` parameter.

Using mailbox pools for incoming connects

If you use the Host Links listener you may start multiple instances of products accepting connects to the same mailbox name on the same DSA node (`SCID`). Each instance must use a different mailbox extension. The instances of the product will be allowed to start execution, register the mailbox name and extension for which they are accepting connects with the listener, and act as a mailbox pool for applications connecting to the mailbox name.

An incoming connect to a mailbox without an extension will be passed to a random instance of the product which is accepting connects to a mailbox with the correct name, and that isn't currently busy handling a session. An incoming connect with a mailbox and extension will only be connected to a product which is accepting connects that match both.

Example:

```
GUFTsrv -id gu1 -li dsa -mn filetran -mx uft1
GUFTsrv -id gu2 -li dsa -mn filetran -mx uft2
GUFTsrv -id gu3 -li dsa -mn filetran -mx uft3
```

The three instances would start, all listening on mailbox `filetran`. Connects to mailbox `filetran` would be given to one of these GUFTsrvs that was not already busy.

Multiple DSA nodes

A single Host Links platform can accept calls from several different DSA nodes. The DSA nodes must be configured in the remote system (Bull Mainframe, Datanet, MainWay, GCOS6 or other UNIX system) that will be issuing the connect request, as separate DSA nodes and transport stations, but using the same network address.

GUFTsrv can use any of these local node names as an argument to the `-LN` parameter to specify a specific node for which the GUFTsrv will accept connects. These local node names used to accept incoming calls can be configured in the DSA configuration of the Host links platform. This is generally just for documentation, but some OSI-stacks require all local node names to be configured. Please refer to the *Host Links Gline manual* for details.

Each GUFTsrv that is accepting incoming calls needs such a local node, and it cannot be shared with other products that accept calls, unless a DSA listener is running on behalf of the node.

Please note that this functionality is not available in Host Links for Windows, or UNIX/Linux if using RFC1006.

Running GUFTsrv via a G&R Ggate gateway

GUFTsrv can be set up to run via a *`G&R/Ggate`* gateway connection. In this case the local GUFTsrv system does not need to have DSA configured, but communicates with the remote DSA host system via a *`Ggate`* process that executes on some other Host Links node in the network (the *`Ggate`* dialog uses TCP/IP transport protocol). Please note the following considerations for such configurations:

‘On demand’ startup is not possible if GUFTsrv is executing over a *`Ggate`* connection. This means that the GUFTsrv process must be pre-started at startup time in order to register its mailbox with the remote Host Links system. This ‘registration’ involves connecting to the *`Ggate`* program and issuing the listen request to the listener there. Please note that once a connection comes in from a UFT requester and a session is established with the remote GUFTsrv program, any other incoming connect for GUFTsrv will be rejected until the (only) active GUFTsrv session has terminated (but the remote UFT requester will normally retry the connection attempt later). This effectively limits such configurations to one UFT session at a given time. Also note that if the GUFTsrv process should ‘hang’ for some reason (e.g. network problems on the TCP/IP connection), no new UFT sessions will be set up until the hanging GUFTsrv process is manually terminated and restarted (but see the description of the ‘idle time ‘ and ‘keep alive’ parameters in the parameter chapter).

GUFTsrv – platform specific issues

GUFTsrv for Windows

All copies of GUFTsrv must run as Windows services, and can only be administered using the *`G&R/Gservice`* program (see a description of the *`Gservice`* program in *the ‘Installation and Configurations on Windows’* manual). For start on demand configurations the DSA listener launches GUFTsrv programs via *`Gservice`*. Copies of GUFTsrv that are to be started at Windows start-up time must be configured in *`Gservice`*’s configuration file. The *`G&R/Gmanager`* utility can be used to monitor active GUFTsrv programs and a command interface is available for stopping active copies of GUFTsrv, restarting copies that have stopped, issuing GUFTsrv operator commands, examining the log files and etc.

GUFTsrv for UNIX/Linux

Status information about the running GUFTsrv processes can be obtained using the UNIX/Linux '`ps`' command. Based on the process_id information from `ps` GUFTsrv, daemons should be stopped using the normal UNIX/Linux `kill` command. All network events are reported to the log in the standard *`Gline`* format with a header identifying the line-handler, e.g. $$DSA from `gl_dsa`.

GUFTsrv parameters

GUFTsrv takes its parameters from the command line only. The following parameters are supported:

Parameter	Description	
`-ID ext`	Mode ID. GUFTsrv identifier and filename extension for configuration file. Maximum 3 characters. When running multiple copies of GUFTsrv each GUFTsrv **must** be started with a unique identifier unless start on demand is used. The default extension is `.dsa` or `.diw` depending on the DSA protocol used.	
`-BIN`	Forces file types to FIX if the requester sets BINARY character set. Necessary with some older host implementations.	
`-LN`	This parameter is optional. It can be used to specify the local DSA node name (`SCID`) to be used by GUFTsrv when listening for incoming connects to its mailbox name. If not set on the command line, this parameter will be given automatically by GUFTsrv and signals that the default local DSA node name will be used for listening. In order to allow other *`Host Links`* communications products that accept connect requests (e.g. *`G&R/Gspool`*, *`G&R/Gmailer`* or other GUFTsrv) to run on the same node, the listener (see description in the *`Gline`* documentation) must be used.	
`-DD data directory path-name`	When the remote UFT requester uses a relative file name, GUFTsrv will look for this file in the current data directory. There is one data directory for each DSA node name (`SCID`) for which connections are being accepted.	
	Windows	`\gar\servers\<scid>.uft`
	UNIX Linux	`/usr/gar/servers/<scid>.uft`

Parameter	Description
	If you want GUFTsrv to look for local files elsewhere, the `-DD` parameter can be used to specify the path of this data directory.
`-DBG`	Enables an internal trace of GUFT events.
`-IT n`	Enable idle timer. Disconnect the session with the requester if no UFT protocol message is received within ‘n’ seconds.
`-KI n`	Enable ‘keep alive’ timer. Applicable when GUFTsrv is running over a Ggate connection. The parameter value (n) is in seconds. If no Ggate keep-alive packet is received in ‘n’ seconds, GUFTsrv disconnects the session with the requester.
`-LK`	Lock file on write. Do not allow other processes to access the local file while GUFTsrv is writing to it.
`-ALK`	Abort lock. Leave the file abort locked if the transfer request is not successfully completed
`-NL`	No logging or accounting. In server mode a log of the performed actions is appended to the GUFTsrv log. The log and account files are written in the standard Host Links directory for server products. You can use this parameter to suppress logging and accounting of server activity.
`-SS`	Single session mode. Terminate after having executed a single transfer request.
`-SZ`	Maximum record size in VAR mode. Fixed record size in FIX mode. Default 512.
`-XL`	Transliterate between 7-bit national characters on the remote system and the ISO/DO11 8 bit equivalents on the local system.
`-UV`	Enable user validation. If this option is set, GUFTsrv will operate on the user ID of the UFT requester user when accessing local files rather than the user ID of the GUFTsrv user.

Parameter	Description
`-CMD cmdline`	Execute 'command line' after a successful transfer request.
`-ECMD cmdline`	Execute 'command line' after any transfer request. GUFTsrv supplies information about filename, status, direction and node names as parameters.
`-TCMD cmdline`	Execute 'command line' when GUFTsrv terminates.
`-SATTR`	Force space attribute information even if the host requester does not ask for it (necessary for some GCOS7 UFT implementations when operating on upper case file names)
Additionally any valid line parameter can be specified as long as it is preceded by:	
`-LI diws/dsa`	See the description of line parameters in the ***Line parameter*** section above.

The GUFTsrv logfile

You can check the results of GUFTsrv activity by looking in the log file.

For every session, the important session parameters and any file transfers are recorded in the log file. The information includes date and time stamp, the calling user mailbox name, the calling node name, the request ID, the local files received or transmitted and the number of bytes transferred. The file can be examined using *`G&R/Gmanager`* or manually with *`Glist`* or any other file list program. An example of a logfile entry follows:

```
[1998/11/20 15:32:21] G&R/Guftsrv 5.2.0b Oct 14 1998 started
[1998/11/20 15:32:21] Connection from: IS2C, user , mbx GUFT
[1998/11/20 15:32:21] Data transfer request, xfer id = 5662015
[1998/11/20 15:32:21] Received 27 chars to local file
/share/mail/gmail/servers/a1
[1998/11/20 15:32:23] Disconnected
```

For more info about logging and accounting see the section entitled *Log and accounting* in the *Administration* chapter, on page 56.

GUFTsrv file system access control

GUFTsrv by default runs on behalf of the local user i.e. the user who started the GUFTsrv process. This implies that any file system access carried out as a result of an incoming request from a remote UFT requester, is performed using the local user's permissions rather than the ones belonging to the remote UFT user. If it is desirable to run on behalf of the remote user rather than the local user, a run time parameter is available (see parameters above) and causes GUFTsrv to validate and use the identity given in the UFT request before any file system activities are performed. Please note that in order for GUFTsrv to be able to change identity, the local user must have administrator (e.g. 'root') types of privileges and this is checked at GUFTsrv startup time.

GUFTsrv redirection

When GUFTsrv receives a file transfer request it checks the local file name given by the requester for redirection format (see a description of the format below). If redirection format is found, GUFTsrv assumes that the file is to be redirected to a client workstation. GUFTsrv then picks up the network address suffix from the file name and opens a socket connection to the GUFTsrv workstation server, GUFTws, which will receive or transmit the requested file. GUFTsrv acts as a gateway moving data between the host requester session and the client workstation.

Starting the workstation redirector server program GUFTws

The workstation redirector server program is delivered as part of the G&R UFT requester package that must be installed on the client workstation. It is started using the command:

```
c:\gar\bin32\guftws.exe
```

The program operates in the background waiting for transfer requests (using TCP/IP port 30865) from GUFTsrv executing on some *Host Links* system. The GUFTws program identifies itself in the system tray on the user's workstation. Once a session is established with a GUFTsrv process and a transfer request is received, the icon will change color and rotates to indicate that a file transfer is in progress and the direction of the transfer (upload/download).

Redirection file name format

A simple private extension of the file name format given in the requester command signals GUFTsrv that the file in question is not a local file but rather a file located on a remote client workstation. The real file name is suffixed by the network address of the client workstation preceded by a semicolon.

The following is an example of using the GCOS7 requester to issue a transfer of a GCOS7 file called 'myfile' to a file called 'myWsFile' on a workstation with the network address 188.99.77.123 via GUFTsrv on DSA address EN01:

```
eftr my.lib..myfile $EN01:'myWsFile;188.99.77.123'
```

GUFTws parameters

Parameter	**Description**
`-DD data directory path-name`	GUFTws will read and write files located in subdirectory `My Received Files` in directory `My Documents`. If you want GUFTws to access local files relative to some other directory, the `-DD` parameter can be used to specify the path of this data directory.
`-DBG`	Enables an internal trace of GUFTws events. The trace file will be called `guftws.dbg` and will be located in the `gar\debug\USER` directory.

Administration

Gmanager

As with all other Host Links server programs, all active GUFTsrv programs for UNIX/Linux or Windows report their current status to the Gmanager database. This can be viewed using Gmanager for UNIX/Linux or Gmanager for Windows. All that is necessary is that the Gmanager that is started has access to the G&R system directory. If the same G&R system directory on a file server is shared by several Host Links platforms, Gmanager will show the status of all Host Links servers on all the Host Links platforms sharing the G&R system directory. Gmanager's server list will show you the most recent status message from all active GUFTsrv programs sharing the G&R system directory, and you can view their log files, see below, at the touch of a key or a mouse. You can also issue commands to GUFT interactively from Gmanager. All active GUFTsrv programs must for this reason have a unique ID (`-ID`).

Commands accepted from Gmanager

Standard commands

The commands that are accepted by all servers are:

- DOWN - terminates the server
- STATUS - reports server-specific status information to the log file
- PARAM - brings up a dialog box that allows the operator to give a command line parameter to the server. Note that some parameters do not work when given interactively i.e. they can only be handled at server startup time
- DEBUG ON/OFF - toggles on and off tracing interactively

GUFT accepts only the standard commands.

Log and accounting

For every session, the important session parameters and any file transfers are recorded in the log file. The information includes date and time stamp, the calling user name, the calling node name, the files received or transmitted, the number of bytes and the throughput rate. The file can be examined using Gmanager or manually with Glist or any other file list program. Additionally GUFT in both server and command modes writes an accounting file that may be used for billing purposes.

GUFT and GUFTw logging

In command mode a summary of the transfer is appended to a log `guft.log` and an accounting file `guft.acc` in the directory where GUFT or GUFTw is started. The format is as described below.

GUFTsrv logging

As with all other Host Links server programs, all active GUFTsrv instances now write an event log in a product specific directory under the `servers` directory in the G&R system directory. There is one GUFTsrv directory for each DSA node name <`SCID`> for which connections are being accepted. The default GUFTsrv directory where the log file is found is:

Windows	`\gar\servers\<scid>.uft`
UNIX/Linux	`/usr/gar/servers/<scid>.uft`

The log file name is `_logfile.<id>` where the `ID` is by default `dsa` or `diw` depending on the DSA protocol used, and which must be made unique if multiple GUFTsrvs are used, by setting the `-id` parameter. When start on demand is used the `IDs` are generated as `_00`, `_01`, `_02` etc.

The event logs are 64k long, and we keep the last three generations for reference.

```
Directory : K:\GMAIL\SERVERS\IS2B.UFT

file.._LOGFILE.DSA
file  _LOG_001.DEF
file  _LOG_002.DEF
file  _LOG_003.DEF
```

Log file format

Example of the content of the log file:

```
File: K:\GMAIL\SERVERS\IS2B.UFT\_LOGFILE.DSA

[1997/03/06 09:49:00] G&R/Guftsrv UFT server 5.0.0 Mar  6 1997
   started
[1997/03/06 09:49:47] Connection from: EN3D, user ARILD, mbx GUFT
[1997/03/06 09:49:47] Data transfer request, xfer id = 3287514605
[1997/03/06 09:49:51] Received 5823 chars to local file
   /local/home/arild/6kb
[1997/03/06 09:50:07] Disconnected
```

The accounting file

In addition to the log file an account file is also produced. This file will be located in the same directory as the log file (system directory for GUFTsrv or start-up directory for GUFT and GUFTw command modes) and contain much of the same information but in a format more suitable for programmatic treatment. The file is `_account` for GUFTsrv or `guft.acc` for command mode. The record contains the following fields separated by tabs (hex 09) and the file can even be read direct into Excel:

Date	**yyyymmdd**
Time	**hhmmss**
r/t	**receive or transmit**
local file name	
remote file name	**only known to GUFT clients**
remote node name	
remote mailbox name	
remote user name	
transmission ID	
# of chars sent/received	
elapsed time in seconds	

The tabs are shown below as semicolons for illustration. Note that the remote file name is not known so the field present, but empty:

```
File:                   K:\GMAIL\SERVERS\IS2B.UFT\_ACCOUNT

19970306;094948;r;/local/home/arild/tst;;EN3D;GUFT;ARILD;328751460
         5
         ;5823;1
```

Troubleshooting

If you are experiencing any kind of problem when using GUFT to transfer files, the GUFT trace file and/or the line handler trace file will provide useful documentation of the problem, either for your own use, for the G&R distributor or for G&R if it turns out to be caused by an error in the program itself. See the appendix *Host Links Trace* for a full discussion of how to generate G&R/Host Links trace files.

GUFT Trace file

This trace file contains details about GUFT processing of host input. To enable this tracing, add the `–DBG` option to the GUFT startup command or to the relevant section of the GUFT configuration file:

```
-USER
      -DBG ON
```

Line handler trace file

This trace file contains details about line handler operation. To enable line handler tracing, add one or both of the `–D_` and `–S_` options to the GUFT start-up command or to the relevant section of the GUFT configuration file:

```
-LI YYY
      -S_ ON
      -D_ ON
```

(YYY =line handler identification, i.e. DSA, DIWS)

GUFT and line handler trace file examples

Examples of directory and file-names

`/usr/gar/debug/mike`	**`Debug directory for user ‘mike’`**	
`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 DSA node ‘en01’**	

`guf.def`	GUFT server debug file	`(-dbg)`
`guf-gli.def`	GUFT server host line trace	`(-li dsa -s_)`

When connecting through Ggate

UNIX/Linux	`/usr/gar/debug/`**`ZZZZ`**`/gga`**`NN-PPPP`**`.dbg`
Windows	`C:\gar\debug\`**`ZZZZ`**`\gga`**`NN-PPPP`**`.dbg`

(ZZZZ = DSA node name, e.g. EN06 or PH13)
(NN =Instance number, starting at 01)
(PPPP =IP-address of the client system, running Gspool in this case)

When GUFT or any other G&R or customer applications based on GlAPI connect through Ggate to another application, the line handler trace will be generated on the Ggate system, with the name and location shown in the table above. In this case the GUFT start-up command or GUFT configuration file would look like this:

```
-LI YYY:PPPP
      -S_ ON
      -D_ ON
```

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

UFT Server-specific information

The GCOS8 UFT

The UFT program under GCOS8 is named DSAS. It has a requester mode and a server mode. The User Master Catalog (UMC) used by DSAS must have read, write and create permission on the UMCs accessed by the GUFT user.

The UMCs accessed must have a catalog under the root named `DSAS_SEC`. Directly under this catalog you need a catalog named identically to the DSA node name from which GUFT will connect. This catalog must again have catalogs with names identical to the user names (`-DU` parameters) to be used by GUFT users. An example: From a GUFT user called `JIM` on a DSA node called `IS52` you want to access a UMC called `MYUMC`. You need the following structure:

```
MYUMC/DSAS_SEC
MYUMC/DSAS_SEC/IS52
MYUMC/DSAS_SEC/IS52/JIM
```

File types that can be created by GUFT under GCOS8 are:

VAR	GFRC ASCII (TSS format)
FIX	GFRC ASCII with fixed length
UFAS sequential	
UFAS relative	

The GCOS8 UFT server will usually not implicitly create a file that you send to it. You must create it using GUFT directives prior to the transmission request. When using the GCOS8 UFT requester, a typical transfer command (after having launched the DSAS program) is:

```
tran -rn xxxx -lf myumc/txtfile -rf ‘/usr/txtfile’ -s -moni -r
```

This command will result in a connect to GUFTsrv on the node `xxxx`, a send (`-s`) of the GCOS8 file `myumc/txtfile`, to the Host Links file system replacing (`-r`) any already existing file by that name. The file transmissions progress will be shown (`-moni`).

The GCOS6 UFT

The GCOS6 UFT facility runs under the MOD 400 operating system in a separate task group with identifier $X. It has a server and requester mode. It is available in two versions; basic and extended. The basic version is sufficient for communication between heterogeneous systems.

Access to files is controlled by means of standard GCOS6 ACLs and CACLs. Creating *VAR* or *UFAS Seq.* type of files results in standard UFAS ASCII files with variable record length. *FIX* files result in UFAS ASCII files with fixed record length (length equal to the selected block length). UFAS Relative files result in UFAS Fixed Relative (bound unit format) files.

The GCOS6 UFT will not implicitly create a file you send to it. You must create it using GUFT directives prior to the transmission request.

The GCOS7 UFT

The GCOS7 requester is GTP (Generalized Transfer Protocol) and is started interactively (from IOF) by means of the EFTR command. It can also be started by the EJR command with JCL containing a FILTFR activity.

The server, GSP (Generalized Server Protocol), is started upon request from a remote UFT requester (i.e. GUFT).

The remote systems must have been defined with an RSYS/RSC pair of directives in the GCOS7 network generation. Additionally they must of course be configured in the system generation if a FEP (DN/MainWay) is used.

The files accessed can be cataloged or non-cataloged. From GUFT the user can send text files (VAR or UFAS seq.), binary files (FIX) and UFAS relative files. If running a GCOS7 UFT prior to V6, file creation and deletion is not available. Any file accessed must be created ('allocated') before it is accessed. Library members will be created automatically though.

A typical GCOS7 UFT requestor command could be:

```
eftr my.lib..txt $xxxx:'/usr/mydir/txt'
```

This will result in a connect from the GCOS7 UFT requester to GUFTsrv on node `xxxx` and the GCOS7 file (member) `my.lib..txt` will be sent to `/usr/mydir/txt` on the Host Links system. The position of the filename on the command line, sets the file transfer direction (the first filename given is the 'source' and the second is the 'target').

The Bull UNIX UFT

The DPX UFT requester is simply called UFT and the server is called UFTS. The requester is started with the UFT command. It will take commands interactively or from a command input file.

The remote systems must have been defined in the `/etc/isohosts` file and the 'node-name' directive must include the service names (i.e., `'NODE' UFTP: 0x10 UFTS: 0x10`).

When answering the 'remote machine type' question from the DPX UFT requester, choose type '5' (other).

The files processed by the DPX UFT are 'byte stream files' (also called 'binary') and sequential files i.e. with variable length records separated by LFs. 'Block mode' is the fastest transfer mode and should be chosen whenever possible.

When connecting to a DPX UFT system, the user name and password used by GUFT/GUFTsrv will be validated. If the user is not known to the DPX system, the connection will be rejected.

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
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 License Keys

All G&R products require a license key to run. If you are a G&R distributor you need a license key from G&R. If you are a customer you should have received the license keys from your distributor together with the software. The licenses are stored in text format in a file named `licenses`.

If `licenses` is delivered with the product files, it is merged with any existing licenses in the configuration directory when you run the install procedure.

UNIX/Linux	`/usr/gar/config/licenses`
Windows server	`C:\gar\config\licenses`

Glicense

The `Glicense` program is included in every software delivery, and it can be used even though no license key is installed. This allows you to create or modify your own licenses from a license card. You must execute `Glicense` from a user-id that has permission to write in the configuration directory (i.e. the Host Links administration user `gar` for Host Links). When executed with no parameters, `Glicense` will check for an existing `licenses` file. If found it will skip directly to the command dialog, but if there is no license then it will prompt you for distributor name, customer name and the main license key. Be careful to type the names and the key exactly as given to you by your distributor. It is important that you respect case and spaces between words.

To tell `Glicense` explicitly where the license file is, or where it should be written, supply the full path as an option. For example:

```
glicense /usr/gar/config/licenses
```

Enter the license information, text and keys, exactly as specified on the supplied license card.

Once the first time installation has been done, you can simply run `Glicense` without any options and it will automatically find the `licenses` file.

When started `Glicense` gives you the following prompt:

```
Enter command or '?':
```

If you enter '?' a list of the available commands is returned:

Use these commands to define/modify and save the `licenses` file.

A = Add	Add a new product to the license file.
D = Delete	Delete a product from the license file.
R = Read	Read in a new license file.
M = Merge	Merge in license keys from a license file
W = Write	Save the license file.
P = Print	Display a list of configured products.
N = New	Create a new license file.
H = Hardware	Change the hardware platform
X = eXpiration	Define an expiration date for all products.
V = Version	Set new version for all products
Q = Quit	Quit the `Glicense` program.
ENTER	The ENTER key quits `Glicense`

If you enter the Print command, the result will be something like this:

```
Enter command or '?': p
Distributor: Bull A/S   Customer: Arbeidsdirektoratet
Product: Basic
Product: Gline
Product: Ggate
Product: Gspool
Product: Qsim
```

License keys

Below you find a complete list of all Host Links and Glink for Java license keys:

License key	Products that require this key
`basic`	All products.
`ggate`	Ggate.
`gspool`	Gspool.
`guft`	GUFT server.
`Guftc`	GUFT client.
`gproxy`	Gproxy.
`qsim`	Qsim.
`v78sim`	V78sim.
`g3270`	G3270.
`g5250`	G5250.
`pthru`	Pthru.
`sdkglapi`	GlAPI SDK.
`Glapi`	GlAPI run-time.
`Telnet`	Enable Ggate Telnet support
`Tnvip`	Enable Ggate TNVIP support
`ssl`	Enables SSL support
`Marben`	Marben OSI Stack for Windows NT 4
`Marben2K`	Marben OSI Stack for Windows 2000
`MarbenXP`	Marben OSI Stack for Windows XP
`Marben03`	Marben OSI Stack for Windows 2003
`Gweb`	Gweb Professional Edition All terminal emulations
`Gljopen`	Glink for Java Open (VTnnn/ANSI/Minitel).
`Gljpro`	Glink for Java Professional Edition.

License key	Products that require this key
`Gljent`	Glink for Java Enterprise Edition.
`Gljall`	Glink for Java All terminal emulations.
`Gljsa`	Glink for Java Standalone (No configuration server).
`Gljsrv`	Glink for Java Server (Configuration and License).
`Gljcnx`	Glink for Java Gconnect (J2EE connector)
`Gljdsa`	Glink for Java DGA (native DSA comms stack)
`Gljggen`	Glink for Java Gargen

Appendix: Host Links Server Administration

Gmanager is the Host Links administration tool. It can be used to control, configure and monitor all the G&R Host Links server programs.

The dialog and interaction between the server programs and Gmanager is based on information located in a database file `_active.srv` that is located in the Host Links `servers` directory. The first time a Host Links server program starts up it registers itself in this 'active' file. Thereafter the server program updates this database with status information whenever the server is active.

The Gmanager program is available in 2 different versions – a Windows GUI based version `gmanw.exe` and a character based subset `gman` (UNIX/Linux binary) or `gman.exe` (PC console application).

The basic functionality of the two versions is the same, but the Windows version interfaces directly to other Windows-only Host Links administrative tools (*`Gconfig`*, *`Gservice`*), and can also start the browser directly to view HTML reports produced by Gproxy, if enabled, or to view the HTML pages associated with a *`Gweb`* or *`Glink for Java`* installation.

The *`Gproxy`* reports, *`Gweb`* and *`Glink for Java`* web pages are of course available to administrators of UNIX/Linux Host Links systems, and can be viewed by starting a browser manually, and connecting to the appropriate URLs:

```
http://mysite.mydomain.com/Gproxy
http://mysite.mydomain.com/Gweb
http://mysite.mydomain.com/GlinkJ
```

A summary of the available functions follows. The Windows-only functions are marked.

Gmanager can be used to perform the most common Host Links administrative tasks i.e.:

- View the last reported status information from the servers
- Start new server
- Restart a server
- Send a command to a server
- View a server log file
- View a server trace file
- Load the DSA configuration into an editor
- Compile the DSA configuration
- Call *`Gconfig`* the server configuration program (Windows)
- `Start the configuration wizard (Windows)`
- Load the *`Gservice`* configuration into an editor (Windows)
- Start the Host Links server programs using *`Gservice`* (Windows)
- Edit the product specific configuration files
- Connect directly to the *`Gproxy`* HTML pages, if enabled (Windows)
- Connect directly to the *`Gweb`* HTML pages, if enabled (Windows)
- View program version numbers, program link information (Windows)
- View license info and license usage (Windows)
- View Host Links environment information, the ‘VMAP’ (Windows)

The commands that are accepted by all servers are:

- DOWN - terminates the server
- STATUS - reports server-specific status information to the log file
- PARAM - brings up a dialog box that allows the operator to give a command line parameter to the server. Note that some parameters do not work when given interactively i.e. they can only be handled at server startup time
- DEBUG ON/OFF - toggles on and off tracing interactively

Additionally, the server in question might support other interactive commands. For a description of the supported commands, check the server-specific documentation.

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/DIWS 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 the `gtsupd` utility e.g.:

```
gtsupd hipp -st enbl
TS-entry ‘hipp’ updated OK. Old state = LOCK, 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\gga`**`NN-PPPP`**`.dbg`
UNIX Linux	`/usr/gar/debug/NODE/gga`**`NN-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.