GX and GSM (Unix)

1. Introduction

The GX client was developed to connect to a GSM (Windows) host. However, the client is not restricted to Windows hosts and can also be connected to Unix servers. This document describes the special considerations that must be followed when connecting GX to a GSM (Unix) host.

The GX.EXE client uses the GXIO.EXE module to provide the interface to the GSM (Windows) or Unix host. It is not surprising, therefore, that the customisation file changes described in this document mainly affect the GXIO.INI file that is used exclusively by GXIO.EXE; and the section of the GXHOSTS.INI file that GX.EXE uses when initiating GXIO.EXE.

The initial connection between GXIO.EXE and a GSM (Unix) host is radically different from the connection between GXIO.EXE and a GSM (Windows) host. When GX connects to a GSM (Windows) host, GXIO.EXE forges an immediate connection to the NETWORK controller within GLOBAL.EXE. The operator-id and password that are (normally) supplied by the operator are passed directly to $AUTH, $PASSWD or $AUTH32 (i.e. to whichever Global authorisation program that has been customised using $CUS).

When GX connects to a GSM (Unix) host, GXIO.EXE connects to the Unix telnetd daemon and interacts with the Unix login processing. There are two fundamentally different techniques to connect a GX client to a GSM (Unix) host. The first technique which is expected to be employed at most live sites, involves customising GXIO.EXE to provide responses to the Unix user name and password prompts; then to supply a command, normally the GSM (Unix) start-up program, *global*, or a GSM start-up script, to the Unix command shell. The second technique, mainly intended for troubleshooting and development, allows GX, via GXIO.EXE, to run as a "dumb" telnet client allowing the Unix user name and password, and dialogue with the Unix command shell, to be entered interactively.

In general, the examples in this document assume a GXHOSTS.INI file that contains entries for a combination of multiple GSM (Windows) servers and multiple Unix servers (i.e. the "HostID" drop down available in the GX Login Dialogue Box will include a mixture of GSM (Windows) and Unix hosts). Some simplifications can be made if the GX client is to be configured to a single Unix server.

2. Customising GX to Login to Unix and Run GSM (Unix)

This section describes the GXHOSTS.INI and GXIO.INI settings that are required to connect GX, via GXIO.EXE, to a Unix host, provide responses to the Unix user name and password prompts, and run a script that executes GSM (Unix).

2.1 Indicating a Unix Host

By default, GX.EXE (and thus, GXIO.EXE) assumes the host server is running GSM (Windows) so the following setting in the Hosts section of GXHOSTS.INI must be enabled to indicate that the target host is a Unix server:

```
UnixHost=On
```

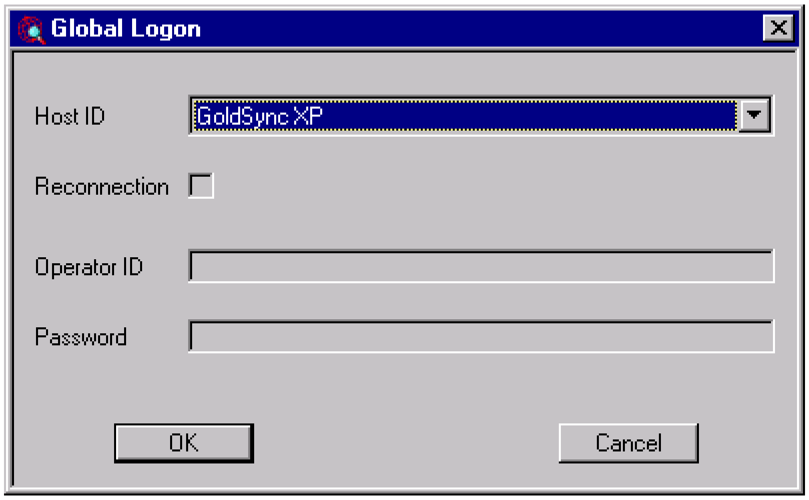
Note that depending on the value of the SeparateHostSections setting, in the [options] section of the GXHOSTS.INI file, this option will appear as:

```
[hosts]
HostID1=name_of_unix_server, description_of_unix_server
UnixHost1=on
```

or, if SeparateHostSections=On, as:

```
[host1]
HostID=name_of_unix_server, description_of_unix_server
UnixHost=on
```

When the HostID for a Unix server is selected the normal GX.EXE sign-on dialogue box that is displayed for Windows servers:



Global Logon

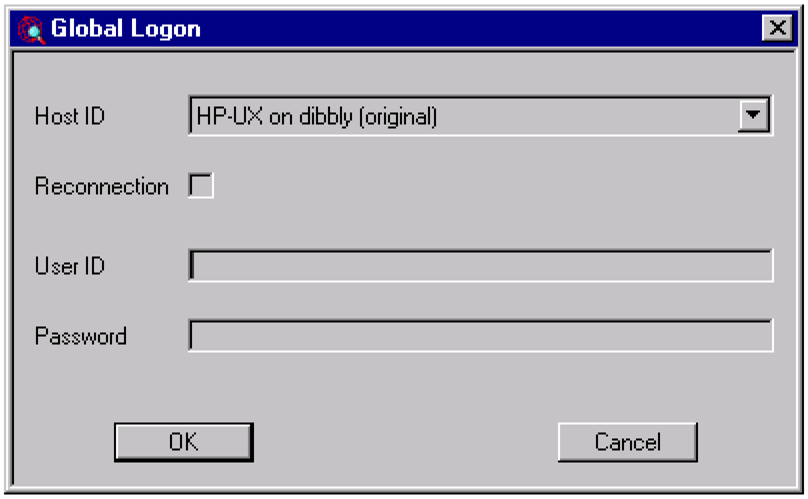
Host ID: GoldSync XP

Reconnection: ☐

Operator ID:

Password:

is replaced by a slightly different dialogue box to support the Unix login dialogue:



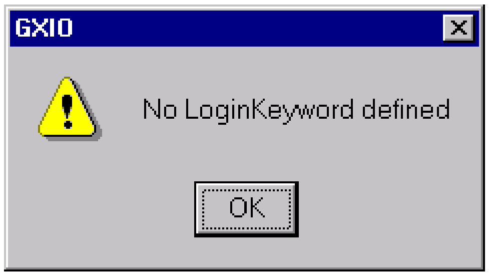
Note that the prompt (Unix) "User ID" has replaced the (GSM (Windows)) "Operator ID" prompt.

Note that the length of the reply to the "Operator ID" edit control in the GSM (Windows) connection window is limited to 4 characters; the length of the reply to the "User ID" in the Unix connection window is essentially unlimited to support the longer Unix user names.

Note also that all the default labels and button text in the GX connection window can be overridden by settings in the GXHOSTS.INI file. For example, the "Password" string can be replaced by the string value of the GXHOSTS.INI "PasswordText" setting.

2.2 Unix Login Procedure

By itself, the UnixHost=On option is not very useful as it merely allows GXIO.EXE to connect as a telnet **client**. By default, GXIO.EXE does not connect as a telnet **emulator** so the connection will hang with the Unix login process waiting for a response which GXIO.EXE will never supply. To avoid this situation, GXIO.EXE will display the following Message Box if a LoginKeyword (see below) is not defined for a Unix connection:


GXIO

No LoginKeyword defined

The responses that GXIO.EXE will provide to the Unix user name and password prompts are specified in the [loginevents] section of the GXIO.INI file.

When the UnixHost=On setting in the GXHOSTS.INI file is specified the initial window displayed by GX.EXE is used to accept the Unix user id and password (i.e. rather than the Global operator-id and password as is the case when GX is connecting to a GSM

(Windows) host). The Unix user id and password are passed to GXIO.EXE from GX.EXE then sent to the Unix host when certain events occur during the Unix login procedure. These events are triggered by GXIO.EXE detecting keywords (or phrases) sent from the host (e.g. login:).

The keyword checking is initiated in two ways:

- After a timeout has expired (counting from the last character received);

- When a <CR> character is received, unless an event action has already been triggered for that line.

Note that the keyword checking only takes place on the current line (hence the need to check on every <CR> character).

The various Unix login events are defined in the GXIO.INI file via settings in the [loginevents] section. Each entry consists of an event keyword (i.e. the string that when detected triggers an event) and an event action which specifies what to do once the associated event has been detected.

There are two predefined events to handle the login and password prompts generated by the Unix host. For these pre-defined "event actions" only the "event keyword" needs to be specified by the LoginKeyword and PasswordKeyword settings respectively. Up to 10 extra "free-format" events can be defined by pairs of settings, EventKeyword% and EventAction% (for % in the range 1 to 10).

The following sections describe the settings in the [loginevents] section of the GXIO.INI file.

2.2.1 LoginKeyword

This setting defines the "event keyword" that triggers the predefined Unix login "event". A typical LoginKeyword setting is "login:". For example:

```
[loginevents]
LoginKeyword=login:
```

The user id string entered on the GX.EXE logon window is passed to GXIO.EXE then sent to the host with a <CR> terminator.

Important Note: If the LoginKeyword setting is not defined in the GXIO.INI file then GXIO.EXE will display an error message and terminate the login attempt.

2.2.2 PasswordKeyword

This setting defines the "event keyword" that triggers the predefined Unix password "event". A typical Password Keyword setting is "password:". For example:

```
[loginevents]
PasswordKeyword=password:
```

The password string entered on the GX.EXE logon window is passed to GXIO.EXE then sent to the host with a <CR> terminator.

Important Note: If the PasswordKeyword setting is not defined in the GXIO.INI file then GXIO.EXE will display an error message and terminate the login attempt.

2.2.3 EventKeyword% & EventAction%

Up to ten extra free-format events can be defined by pairs of settings, EventKeyword% and EventAction% (for % in the range 1 to 10). These settings allow extra dialogue generated by the Unix system to be handled. For example, an incorrectly entered password may result in a message "Login incorrect" from Unix. This phrase could then be used as a keyword and the associated action could display a message on the GX logon window to give the user some information. The action is specified in the following way:

```
Command,parameter
```

Two commands are supported, Send and Display. The Send command transmits the parameter string as it stands, except the ':' character which signifies a <CR>. The Display command passes the parameter string to the GX logon window which displays it in a Window message box. For example:

```
[loginevents]
EventKeyword1=Press return
EventAction1=Send,:
EventKeyword2=Login incorrect
EventAction2=Display,Login incorrect
```

Important Note: The event keywords and actions must be entered in pairs, otherwise they will be ignored.

2.2.4 MonitorLoginProcedure

This setting is obsolete and has been superceded by the diagnostic options described in section 2.5.

2.2.5 CaseSensitiveKeywordComparison

This setting specifies whether the keywords entered should be compared as they are (i.e. the keyword must match the displayed prompt/message exactly) or whether a case insensitive comparison should take place (in fact all keywords/message are converted to upper case before the comparison takes place).

The default setting is Off.

2.2.6 SingleUseLoginKeyword

This advanced option is reserved for future use.

2.2.7 An Example [loginevents] section

The following [loginevents] section has been used to login to a SCO Unix system:

```
[loginevents]
MonitorLoginProcedure=On
CaseSensitiveKeywordComparison=On
LoginKeyword=login:
PasswordKeyword=Password:
EventKeyword1=Press return
EventAction1=Send,:
EventKeyword2=Login incorrect
EventAction2=Display,Login incorrect
```

Note that the login and password keywords are set to the **exact** string displayed by the Unix start-up/login script (including the ':' character).

There are two additional events defined. The first event traps a prompt which stops the login procedure just before the user's .profile is executed. This prompt is actioned by sending a single <CR> which allows the login process to continue. The second additional event traps the condition where an invalid password has been entered and the action displays a message on the GX logon window. This is especially important if the GXIO window can't be seen to provide the user information about why the login has failed.

Note also that the case sensitive comparison has been enabled (CaseSensitiveKeywordComparison=On) because the particular Unix server that this [LoginEvents] section was written for, displays a text message containing the string "LOGIN:" after the password has been accepted. The text "LOGIN:" (or "login:") would trigger the login event again and so a spurious Unix user-id would be sent to the Unix host by GXIO.EXE. Specifying a case sensitive comparison means that no occurrences of the text-string "LOGIN:" (upper-case) displayed by the Unix host after the password has been accepted are incorrectly interpreted by GXIO.EXE as the LoginKeyword (i.e. "login:").

2.3 Running a GSM (Unix) Session (BACNAT < V3.317)

Unless further action is taken, the entries in the [loginevents] section of the GXIO.INI file, if successful, allow GXIO.EXE (and thus GX.EXE) to negotiate the Unix login script by supplying the user name and password. It is normally necessary to configure GXIO.EXE and/or Unix to automatically run the GSM (Unix) start-up program i.e. *global*.

In the simple case, where the Unix user is always expected to run a GX session, an entry in the Unix login script (e.g. .profile) to run global is all that is required. For example:

```
GLTERM=gx;export GLTERM
global -x
```

This example illustrates two very important points. Firstly, the global -x command line option **MUST** be enabled for all GX users. This option is required to disable XON/XOFF flow-control characters that could otherwise interfere with the GX protocol.

Secondly, the GSM (Unix) kernel must aware that GX, rather than a "normal" telnet emulator (e.g. TELNET.EXE or GSMWIN32.EXE) is being used, otherwise text-strings, instead of GX command blocks, will be sent to the GX client. Such text-strings will appear in the GX Window-Zero text emulator window. The GX Window-Zero processor cannot interpret "foreign" escape sequences (e.g. ANSI control commands) so, if the text-strings sent by the Unix host, include ANSI commands, "garbage" will appear in the GX Window-Zero emulator window.

For BACNAT V3.316, or earlier, a GX session is indicated by setting the GSM terminal type (i.e. TAP number) to 911. Usually, this is achieved by setting a GLTERM variable in the start-up script to "gx" and mapping the Unix terminal name "gx" to a Global terminal code of 911 in the GSM (Unix) Systems file. Although using GLTERM (or TERM) and a Systems file mapping is the conventional method, other techniques (e.g. setting the GLTT variable to 911) can be used to establish 911 as the terminal type.

2.3.1 Mixing GX and non-GX sessions

In the simple example described above, where a GX session is indicated by the Unix login script (e.g. by the line GLTERM=gx;export GLTERM), the Unix user is always expected to connect to the Unix server via GX. If a user attempts to connect to the Unix server via a "normal" telnet emulator (e.g. TELNET.EXE or GSMWIN32.EXE) the start-up script will still indicate a GX session (by setting the GLTERM (or TERM) Unix shell variable to "GX"; or using the global -f command line option). Thus, the GSM (Unix) kernel will still assume a GX client using 911 as the terminal type. Terminal type 911 will result in GX commands being sent to the non-GX terminal emulator which will normally result in the display of "garbage".

The following technique illustrates how a Unix start-up script, that includes some decision-making logic, can be used in conjunction with an EventKeyword/EventAction pairing in the [LoginEvents] section of the GXIO.EXE file to allow both GX and non-GX sessions to login with the same Unix user-name.

For BACNAT V3.316, or earlier, the script should be extended to the following:

```
unset GLTERM
echo "Key GX or <CR> for telnet emulator:\c"
read JUNK
if test "$JUNK" = "GX"
then
      GLTERM=gx;export GLTERM
fi
global -x
```

The [LoginEvents] section of the GXIO.INI file should contain a pair of settings to supply the response "GX" when the string " Key GX or <CR> for telnet emulator:" is detected. For example:

```
[loginevents]
EventKeyword3=Key GX or <CR> for telnet emulator:
EventAction3=Send,GX:
```

For BACNAT V3.317, or later, (see below) the script could be simplified to:

```
echo "Key GX or <CR> for telnet emulator:\c"
read JUNK
if test "$JUNK" = "GX"
then
      global -f -x
else
      global -x
fi
```

2.4 Running a GSM (Unix) Session (BACNAT V3.317, and later)

A much improved technique to specify a GX connection is available for GSM (Unix) BACNAT V3.317, and later: A GX session can be indicated using the global -f command line option. For example:

```
global -f -x
```

Note that the -x, as described above, **must** be used in conjunction with the -f option. Note also if the -f option is used, the GSM Terminal Type is assumed to be 911 so that there is no need to establish GLTERM, or any related Unix shell variables.

Use of a "clean" command line argument to indicate a GX connection avoids the need for a special "GSM start-up" script to set the GSM terminal type to 911 before starting GSM (Unix).

2.5 Diagnostics

Establishing a GXIO.INI [loginevents] section that conforms exactly to the login dialogue for a particular Unix server can often be a painstaking process. Two types of diagnostics are available in GX.EXE/GXIO.EXE to trouble-shoot this procedure.

2.5.1 Enabling the Diagnostic Message Display Window

A DisplayMessages setting in the GXHOSTS.INI file must be enabled in order for GX to display any diagnostics. The DisplayMessages setting can appear in several sections of the GXHOSTS.INI file.

A generic DisplayMessages option in the [options] section of GXHOSTS.INI allows the Diagnostic Message Display Window to be created for **all** hosts.

If it is required to only invoke a Diagnostic Message Display Window selectively for **some** hosts, the DisplayMessages setting in the [options] section of GXHOSTS.INI should be absent, or set to "Off" and a host-specific DisplayMessages% setting enabled on a per-host basis. Note that the generic DisplayMessages setting in the [options] section takes precedence over the host-specific DisplayMessages options in the [hosts] section.

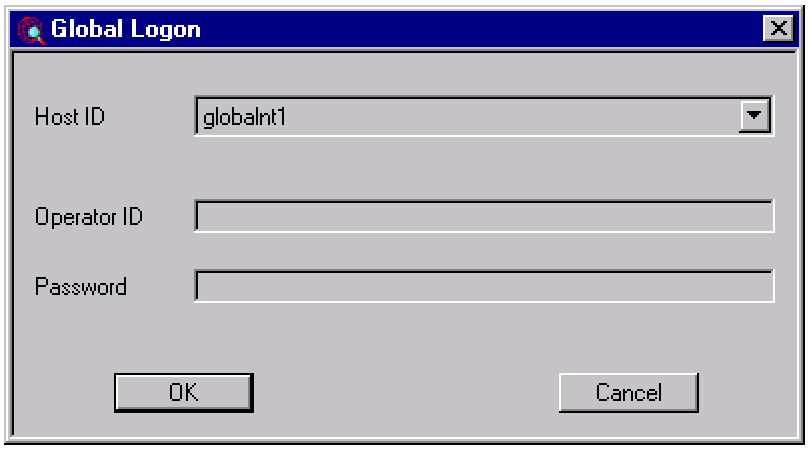
As with all host-specific settings in the GXHOSTS.INI file, the exact format of the DisplayMessages setting depends on the value of the SeparateHostSections setting. For example, if SeparateHostSections=Off:

```
[hosts]
HostID1=unix1, description_of_unix1_server
UnixHost1=on
DisplayMessages1=On
HostID2=unix2, description_of_unix2_server
UnixHost2=on
DisplayMessages2=On
```

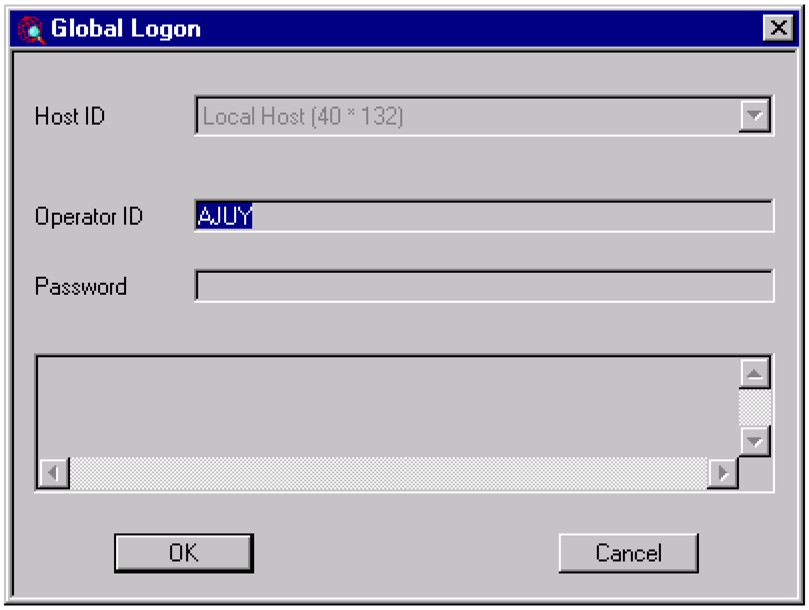
or, if SeparateHostSections=On, as:

```
[host1]
HostID=unix1, description_of_unix1_server
UnixHost=on
DisplayMessages=On
 [host2]
HostID=unix2, description_of_unix2_server
UnixHost=on
DisplayMessages=On
```

The host-specific DisplayMessages% setting only applies when the HostID% setting, that is equivalent to the DisplayMessages% setting has been selected. The Diagnostic Message Display Window only appears when that host is selected so that the **initial** GX logon dialogue box appears as:



and only switches to the dialogue box with a scrolling text area once that specific host has been selected:



The MessageDisplayHeight setting in the [options] section of GXHOSTS.INI can be used to control the size of the scrolling Diagnostic Message Display Window.

2.5.2 Displaying the GXIO.EXE to Unix Host Dialogue

By default, when the DisplayMessages setting in the GXHOSTS.INI file is enabled the Diagnostic Message Display Window will contain the text strings sent to GXIO.EXE by the Unix server. This option is extremely useful when trouble-shooting problems with the [loginevent] settings described in section 2.2.

2.5.3 Displaying the GX.EXE to GXIO.EXE Protocol Messages

The DisplayMessages settings described above can be used in conjunction with the DisplayLogMessages setting in the [miscellaneous] section of the GXIO.INI to display details of the hand-shaking that takes place between GX.EXE and GXIO.EXE. In general, these internal diagnostics are more difficult to analyse then the textual diagnostics described in section 2.5.2 but are useful for some types of problem.

To summarise the diagnostic options:

GXHOSTS.INI DisplayMessages	**GXIO.INI DisplayLogMessages**	**Diagnostics**
Off	Off	Diagnostics Display Window suppressed
Off	On	Diagnostics Display Window suppressed
On	Off	Diagnostics Display Window contains textual messages sent to GXIO.EXE from the Unix

		host.
On	On	Diagnostics Display Window contains details of the GX.EXE to GXIO.EXE protocol.

2.6 Connecting to Multiple Unix Servers

It is very unlikely, that two different Unix servers will provide exactly the same login dialogue to GX.EXE/GXIO.EXE. Thus, the settings in the [loginevents] section of a single GXIO.INI file are normally specific for a single Unix host. Configuring a GX PC to provide multiple login capabilities to two, or more, Unix hosts **could** be achieved by installing separate GX.EXE/GXIO.EXE, and thus separate GXHOSTS.INI and GXIO.INI etc. files, in different Windows folders. However, installing multiple copies of the same .EXE's in two, or more, folders on a PC is not normally considered "good practice" as it complicates upgrade procedures.

The GXIOINIFILE setting in the [hosts] section of the GXHOSTS.INI file can be used to override the default GXIO.INI file and thus provide each connection to a Unix host with a different set of [loginevents] settings. For example:

```
[hosts]
HostID1=unix1, description_of_unix1_server
UnixHost1=on
GXIOINIFILE1=gxio_unix1.ini
HostID2=unix2, description_of_unix2_server
UnixHost2=on
GXIOINIFILE2=gxio_unix2.ini
```

or, if SeparateHostSections=On, as:

```
[host1]
HostID=unix1, description_of_unix1_server
UnixHost=on
GXIOINIFILE=gxio_unix1.ini
[host2]
HostID=unix2, description_of_unix2_server
UnixHost=on
GXIOINIFILE=gxio_unix2.ini
```

It is not normally necessary to specify different GXIO.INI files for a GXHOSTS.INI file configuration that includes multiple Windows servers.

The [loginevents] section of the GXIO.INI file is only considered for connections to Unix servers (i.e. the section is ignored for connections to Windows hosts). Consequently, it is not normally necessary to specify different GXIO.INI files for a GXHOSTS.INI file configuration that includes multiple Windows servers and a **single** Unix server.

3. Customising GX as a Simple Telnet Emulator

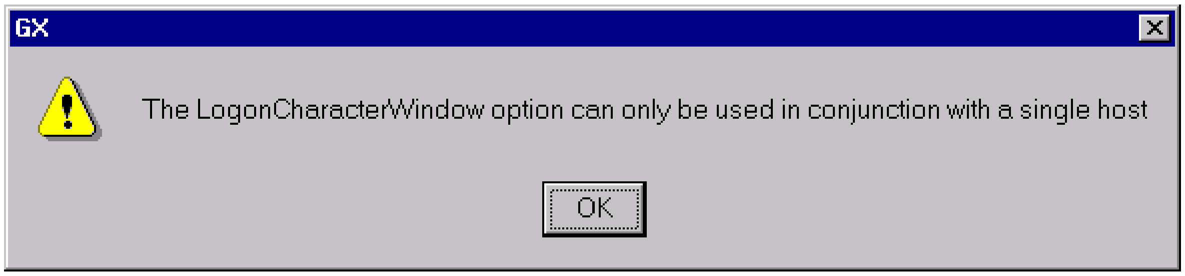
Although the use of [loginevents] settings to sign a GX user into Unix and from the Unix login to run GSM (Unix) it is sometimes necessary to log onto Unix "by hand". GX supports an option to function as a simple telnet emulator i.e. to allow the operator to interact directly with the Unix login script and shell.

Important Note: Although GX functions as a simple telnet emulator it does not support all the escape sequence handling to run "full screen" Unix utilities such as vi.

To configure GX.EXE to operate as a simple telnet emulator the following setting in the [options] section of the GXHOSTS.INI file must be enabled:

```
[options]
LogonCharacterWindow=On
```

Important Note: This option can only be used when the GXHOSTS.INI file only contains a single HostID setting otherwise the following message box will be displayed when running GX:



The LogonCharacterWindow option can only be used in conjunction with a single host

This restriction of a single HostID entry in the GXHOSTS.INI file when the LoginCharacterWindow option is enabled is a real problem if is required to connect GX to multiple hosts from a single PC. The restriction can be overcome by installing, or simply copying, multiple copies of GX into separate folders (i.e. to obtain a series of separately editable GXHOSTS.INI files, each of which contains a single HostID). However, installing multiple copies of the same .EXE's in two, or more, folders on a PC is not normally considered "good practice" as it complicates upgrade procedures. This can be avoided by using the GX.EXE /E command line option to override the default GXHOSTS.INI file. For example:

```
F:\GX\GX.EXE /E=gxhosts-unix1.ini
F:\GX\GX.EXE /E=gxhosts-unix2.ini
F:\GX\GX.EXE /E=gxhosts-unix3.ini
```

where gxhosts-unix1.ini, gxhosts-unix2.ini and gxhosts-unix3.ini are separate GX Hosts files each of which contains a single HostID setting and thus, can be configured with the LogonCharacterWindow=On setting.