

Informix Product Family  
Informix Genero  
Version 2.41

*IBM Informix Genero Desktop Client  
User Guide*





Informix Product Family  
Informix Genero  
Version 2.41

*IBM Informix Genero Desktop Client  
User Guide*



**Note**

Before using this information and the product it supports, read the information in "Notices" on page B-1.

**Edition Notice**

This edition applies to version 2.41 of IBM Informix Genero (product number 5725-D13) and to all subsequent releases and modifications until otherwise indicated in new editions.

Licensed Materials - Property of IBM.

All rights reserved. All information, content, design, and code used in this documentation may not be reproduced or distributed by any printed, electronic, or other means without prior written consent of Four Js Development Tools.

© Four Js Development Tools 1995, 2012



---

# Contents

<b>Introduction</b> . . . . .	<b>ix</b>
About this publication . . . . .	ix
What's new in Informix Genero Desktop Client, v 2.4 . . . . .	ix
Contacting IBM Software Support . . . . .	x
<b>Chapter 1. General</b> . . . . .	<b>1-1</b>
What is the Genero Desktop Client . . . . .	1-1
Starting and Configuring the GDC . . . . .	1-1
Starting the GDC . . . . .	1-1
Configuring the GDC . . . . .	1-1
The Preferences configuration panel . . . . .	1-2
The Advanced configuration panel: Advanced options . . . . .	1-4
The Connection configuration panel: Connection options . . . . .	1-8
The Security configuration panel: Security options . . . . .	1-9
The Report configuration panel: Report to Printer options . . . . .	1-10
Applying a second configuration file . . . . .	1-10
Legal notices . . . . .	1-10
<b>Chapter 2. Applications</b> . . . . .	<b>2-1</b>
Shortcut System . . . . .	2-1
Shortcut System Overview . . . . .	2-1
Administration Mode . . . . .	2-2
Shortcut Management . . . . .	2-2
Starting a Shortcut (Application) . . . . .	2-3
Local Shortcuts . . . . .	2-3
Shortcut Wizard . . . . .	2-4
Starting the Shortcut Wizard . . . . .	2-4
Direct Connection Shortcuts . . . . .	2-4
SSH Tunneling . . . . .	2-7
Local Connection Shortcuts . . . . .	2-7
Connections via the Genero Application Server . . . . .	2-8
Shortcuts and environment variables . . . . .	2-9
Customizing your Login Box . . . . .	2-9
The Connections Panel . . . . .	2-11
The Terminals Panel . . . . .	2-13
The Debug Panel and Logging System . . . . .	2-14
Using the Logging System . . . . .	2-15
Viewing the Debug Console . . . . .	2-16
<b>Chapter 3. Features</b> . . . . .	<b>3-1</b>
The Command Line . . . . .	3-1
Command line options . . . . .	3-1
Command line examples . . . . .	3-5
Printing a screen shot . . . . .	3-5
Local actions . . . . .	3-6
Local actions overview . . . . .	3-6
List of local actions . . . . .	3-7
Localization encoding list . . . . .	3-8
<b>Chapter 4. Upgrading and New Features</b> . . . . .	<b>4-1</b>
Genero Desktop Client 2.40 New Features . . . . .	4-1
Genero Features: Genero Desktop Client 2.40 New Features . . . . .	4-1
Summary Line Support . . . . .	4-1
Built-in search and fast seek . . . . .	4-2

Widgets: Genero Desktop Client 2.40 New Features . . . . .	4-2
ComboBox autocompletion timer . . . . .	4-2
ComboBox completer . . . . .	4-2
Table line decoration review . . . . .	4-3
Web Component: Debugging information . . . . .	4-3
Web Component: gzip support. . . . .	4-4
Web Component: cookies support. . . . .	4-4
Traditional Mode: Genero Desktop Client 2.40 New Features . . . . .	4-5
Global toolbar is now supported . . . . .	4-5
Status bar support . . . . .	4-5
MDI Child application can be a traditional application. . . . .	4-6
"Pop-Tree" StartMenu support . . . . .	4-7
Shortcut mechanism: Genero Desktop Client 2.40 New Features . . . . .	4-7
"Start a new shell" is no more mandatory for auto port forwarding . . . . .	4-7
Monitor: Genero Desktop Client 2.40 New Features. . . . .	4-7
Debug console configuration . . . . .	4-8
Tcp server configuration . . . . .	4-8
Miscellaneous: Genero Desktop Client 2.40 New Features. . . . .	4-9
Report Viewer: improved performances. . . . .	4-9
Genero Desktop Client 2.40 New Features . . . . .	4-9
Genero Desktop Client 2.32 New Features . . . . .	4-9
Genero Desktop Client 2.30 New Features . . . . .	4-9
Genero Features: Genero Desktop Client 2.30 New Features . . . . .	4-9
Drag & Drop support. . . . .	4-9
WebComponent support . . . . .	4-9
Widgets: Genero Desktop Client 2.30 New Features . . . . .	4-10
Frozen columns for tables . . . . .	4-10
Text alignment in headers of tables . . . . .	4-11
DateEdit: include support . . . . .	4-11
Textedit / Spell Checker: Add to dictionary . . . . .	4-11
Adapt window to new form size . . . . .	4-12
Window: Message nodes support styles . . . . .	4-12
IgnoreMinimizeSetting style attribute . . . . .	4-13
SpinEdit: Support of leading 0 format . . . . .	4-13
ComboBox: shortcut key for the NULL item . . . . .	4-13
Monitor: Genero Desktop Client 2.30 New Features . . . . .	4-13
Fgltty based on qPutty . . . . .	4-13
Automatic Port Forwarding . . . . .	4-14
GDC information for better support . . . . .	4-16
Miscellaneous: Genero Desktop Client 2.30 New Features . . . . .	4-17
Settings for REPORT TO PRINTER (in case of DBPRINT=FGLSERVER). . . . .	4-17
Animated Gifs support . . . . .	4-17
WinMAIL: Specific Port for smtp . . . . .	4-17
WinDDE: handling of ASCII/Wide char data . . . . .	4-17
Genero Desktop Client 2.22 New Features . . . . .	4-18
Widgets: Genero Desktop Client 2.22 New Features . . . . .	4-18
Rich Text Editing . . . . .	4-18
Genero Desktop Client 2.21 New Features . . . . .	4-20
General category: Genero Desktop Client 2.21 New Features . . . . .	4-20
GDC 2.21 is now compatible with FGL 2.11 Runtime . . . . .	4-20
Internal HTTP stack has been rewritten . . . . .	4-21
Windows: Genero Desktop Client 2.21 New Features . . . . .	4-22
Automatic scrollbars. . . . .	4-22
A window can be started as minimized . . . . .	4-22
Widgets: Genero Desktop Client 2.21 New Features . . . . .	4-22
DateEdit presentation style showCurrentMonthOnly . . . . .	4-22
SpinEdit new attributes. . . . .	4-23
Image supports alignment attribute. . . . .	4-23
Window supports toolBarDocking . . . . .	4-23
Presentation Styles: Genero Desktop Client 2.21 New Features. . . . .	4-23
Non-integer point font size . . . . .	4-23

Monitor: Genero Desktop Client 2.21 New Features . . . . .	4-23
Connection tab displays application title . . . . .	4-23
User limit exceeded feedback possible . . . . .	4-23
Debugging information added . . . . .	4-24
Windows 7 support . . . . .	4-24
Bitwise WinSSHd 5 support . . . . .	4-24
FrontEnd functionCall: Genero Desktop Client 2.21 New Features . . . . .	4-24
hardcopy frontCall available . . . . .	4-24
launchurl frontCall signature . . . . .	4-24
Miscellaneous: Genero Desktop Client 2.21 New Features . . . . .	4-24
Hardcopy available for MDI child windows . . . . .	4-24
Windows now uses MSI installer system . . . . .	4-25
Improved Desktop integration . . . . .	4-25
Experimental features: Genero Desktop Client 2.21 New Features. . . . .	4-25
Flash support . . . . .	4-25
Compositing . . . . .	4-25
Windows Vista introduced a Command Link Button . . . . .	4-26
Save and restore current window size . . . . .	4-26
Genero Desktop Client 2.20 New Features . . . . .	4-26
General Category: Genero Desktop Client 2.20 New Features . . . . .	4-27
Internal library used is now Qt4 . . . . .	4-27
SVG image format added to supported image format list . . . . .	4-27
Image Cache . . . . .	4-27
Monitor / Shortcut mechanism: Genero Desktop Client 2.20 New Features . . . . .	4-28
A customized login box can be created for shortcuts . . . . .	4-28
Command Line option for ignore settings. . . . .	4-29
Read-Only Stored settings . . . . .	4-29
Command Line Option for recording log . . . . .	4-29
SSH2 is default protocol type when -T is not set . . . . .	4-29
Import / Export shortcut and .gdc files . . . . .	4-29
Security Level in http . . . . .	4-29
Miscellaneous . . . . .	4-30
FrontEnd functionCall: Genero Desktop Client 2.20 New Features . . . . .	4-30
functionCall Standard "getwindowid" added . . . . .	4-30
New parameters for functionCall Standard "feinfo" added. . . . .	4-30
Widgets: Genero Desktop Client 2.20 New Features . . . . .	4-30
Treeview widget support . . . . .	4-30
Labels and TextEdits: Hyperlinks with html textFormat . . . . .	4-31
TextEdit: Integrated search facility . . . . .	4-31
TextEdit: Spell Checker . . . . .	4-31
Folders: StyleAttribute for position . . . . .	4-33
Action Frame (Action Panel and Ring Menu): StyleAttribute for decoration . . . . .	4-33
Menu: Style Attribute for popup menu position . . . . .	4-33
Menu: Menu and mouse wheel . . . . .	4-33
Button: ButtonType style attribute to customize Buttons . . . . .	4-33
Tables: Genero Desktop Client 2.20 New Features . . . . .	4-33
Tables: MultiSelection in DISPLAY ARRAY . . . . .	4-33
Tables: Sort in INPUT ARRAY . . . . .	4-34
Tables: Picture Flow design . . . . .	4-34
Tables: resizeFillsEmptySpace styleAttribute . . . . .	4-34
Windows: Tabbed MDI . . . . .	4-34
Action mechanisms: Genero Desktop Client 2.20 New Features . . . . .	4-35
Local Action priority. . . . .	4-35
Qualified Local Actions. . . . .	4-35
Experimental features: Genero Desktop Client 2.20 New Features. . . . .	4-35
Look and Feel . . . . .	4-35
Form Layout . . . . .	4-36
Integrated Browser . . . . .	4-36
Genero Desktop Client 2.2x migration guide. . . . .	4-38
Genero Desktop Client 2.21 migration guide. . . . .	4-38
Genero Desktop Client 2.20 migration guide. . . . .	4-40

Genero Desktop Client 2.3x migration guide . . . . .	4-43
Genero Desktop Client 2.30 migration guide . . . . .	4-43
Genero Desktop Client 2.4x migration guide . . . . .	4-45
Genero Desktop Client 2.40 migration guide . . . . .	4-45
<b>Chapter 5. ActiveX . . . . .</b>	<b>5-1</b>
Genero Desktop Client ActiveX . . . . .	5-1
Genero Desktop Client ActiveX Overview . . . . .	5-1
Installing Genero Desktop Client ActiveX . . . . .	5-1
Uninstalling Genero Desktop Client ActiveX . . . . .	5-1
Genero Desktop Client ActiveX and the Genero Application Server . . . . .	5-1
Genero Desktop Client ActiveX and the Genero Application Server overview . . . . .	5-2
Installing Genero Desktop Client ActiveX with the Genero Application Server . . . . .	5-2
The Genero Desktop Client ActiveX demo . . . . .	5-2
Genero Desktop Client ActiveX installation issues and solutions . . . . .	5-2
Genero Desktop Client ActiveX template . . . . .	5-2
Template Definition . . . . .	5-3
Genero Desktop Client ActiveX API . . . . .	5-5
<b>Chapter 6. Security . . . . .</b>	<b>6-1</b>
Security levels . . . . .	6-1
Security level 0 . . . . .	6-1
Security level 1 . . . . .	6-1
Security levels 2 and 3 . . . . .	6-1
Security Connection Message . . . . .	6-2
GDC and SSH . . . . .	6-3
GDC and SSH overview . . . . .	6-3
GDC and SSH prerequisites . . . . .	6-4
GDC and SSH simple setup . . . . .	6-5
Port Forwarding and Firewalls . . . . .	6-6
Port forwarding . . . . .	6-6
Port forwarding and the client-side firewall . . . . .	6-12
Port forwarding and the server-side firewall . . . . .	6-16
Implementing a Secure Server with GDC . . . . .	6-22
Prerequisites . . . . .	6-23
Solutions overview . . . . .	6-23
The shell script . . . . .	6-24
Setup SSH login . . . . .	6-25
Setup telnet . . . . .	6-29
Password management . . . . .	6-31
Handling expired passwords . . . . .	6-31
Changing passwords . . . . .	6-32
AUTOPORTFIND source code example . . . . .	6-33
Login script . . . . .	6-37
SSH Configuration Troubleshooting . . . . .	6-38
Duplicate port forward number . . . . .	6-38
Wireless systems . . . . .	6-39
Need to change the port that GDC listens on . . . . .	6-40
Sessions expiring . . . . .	6-40
GDC and Windows XP Service Pack 2 . . . . .	6-40
GDC and Windows XP Service Pack 2 Firewall Configuration . . . . .	6-40
GDC ActiveX and Windows XP Service Pack 2 . . . . .	6-42
GDC and Windows Vista . . . . .	6-42
User Account Control . . . . .	6-43
Genero Desktop Client installation on Windows Vista . . . . .	6-43
Running the Genero Desktop Client on Windows Vista . . . . .	6-43
Genero Desktop Client features affected by the User Account Control . . . . .	6-45
Configuration . . . . .	6-45
File Transfer . . . . .	6-45
Windows Vista and Genero Desktop Client ActiveX . . . . .	6-46

Genero Desktop Client ActiveX installation on Windows Vista . . . . .	6-46
Running the Genero Desktop Client ActiveX on Windows Vista . . . . .	6-47

**Chapter 7. Front End Extensions . . . . . 7-1**

Using Front End Extensions . . . . .	7-1
How Front End Extensions work . . . . .	7-1
The Initialize and Finalize functions . . . . .	7-2
Front End Extension function prototype . . . . .	7-2
Front End Interface structure . . . . .	7-2
Front End environment variables . . . . .	7-3
Front End example: create a basic extension . . . . .	7-4
Windows DDE Support . . . . .	7-5
Using the DDE API . . . . .	7-5
The DDE API function list . . . . .	7-6
DDEConnect . . . . .	7-7
DDEExecute . . . . .	7-7
DDEFinish . . . . .	7-8
DDEFinishAll . . . . .	7-8
DDEError . . . . .	7-8
DDEPeek . . . . .	7-8
DDEPoke . . . . .	7-9
WinDDE example . . . . .	7-10
Windows COM Support . . . . .	7-10
Using the COM API . . . . .	7-11
The COM API function list . . . . .	7-11
CreateInstance . . . . .	7-11
CallMethod . . . . .	7-12
GetProperty . . . . .	7-12
SetProperty . . . . .	7-12
GetError . . . . .	7-13
ReleaseInstance . . . . .	7-13
WinCOM examples . . . . .	7-13
Wincom and Excel example . . . . .	7-13
Wincom and Word example . . . . .	7-14
Wincom and Outlook example . . . . .	7-15
Wincom and Internet Explorer example . . . . .	7-16
Windows Mail extension . . . . .	7-17
The WinMail API . . . . .	7-18
Init . . . . .	7-18
Close . . . . .	7-18
SetBody . . . . .	7-19
SetSubject . . . . .	7-19
AddTo, AddCC, AddBCC . . . . .	7-19
AddAttachment . . . . .	7-19
SendMail . . . . .	7-20
GetError . . . . .	7-20
SetSmtp . . . . .	7-21
SetFrom . . . . .	7-21
WinMail examples . . . . .	7-21
Mail using MAPI . . . . .	7-21
Mail using SMTP server . . . . .	7-22

**Chapter 8. General terms . . . . . 8-1**

**Chapter 9. Security terms . . . . . 9-1**

**Appendix. Accessibility . . . . . A-1**

Accessibility features for IBM Informix products . . . . .	A-1
Accessibility features . . . . .	A-1
Keyboard navigation . . . . .	A-1

Related accessibility information . . . . .	A-1
IBM and accessibility. . . . .	A-1
<b>Notices . . . . .</b>	<b>B-1</b>
Trademarks . . . . .	B-3

---

## Introduction

This publication contains information about IBM® Informix® Genero®.

---

### About this publication

This publication describes how to administer and use the IBM Informix Genero Desktop Client.

The Genero Desktop Client (GDC) is a graphical Front End for the Genero Runtime System.

---

### What's new in Informix Genero Desktop Client, v 2.4

This publication includes information about new features and changes in existing functionality.

The following changes and enhancements are relevant to this publication.

Version 2.41 is a maintenance release. There are no new features for Version 2.41.

The following table lists the new features for Version 2.40.

*Table 1. What's new in Informix Genero Desktop Client, v 2.40*

Overview	Reference
Support for Summary Line introduced.	See "Summary Line Support" on page 4-1
Support for built-in search and fast-peek features introduced.	See "Built-in search and fast seek" on page 4-2
ComboBox autocompletion timer introduced, specifying the delay before an autocompletion search starts new.	See "ComboBox autocompletion timer" on page 4-2
ComboBox style comboboxCompleter introduced.	See "ComboBox completer" on page 4-2
Table line decoration applied across the entire row, even when the columns stop before the allocated space for the table is filled.	See "Table line decoration review" on page 4-3
New messages are shown in the debug console to assist with WebComponent debugging.	See "Web Component: Debugging information" on page 4-3
WebComponent now accepts gzip encoding	See "Web Component: gzip support" on page 4-4
WebComponent now accepts cookies.	See "Web Component: cookies support" on page 4-4
Global toolbar is now supported for applications displayed in traditional mode.	See "Global toolbar is now supported" on page 4-5
Status bar is now supported for applications displayed in traditional mode.	See "Status bar support" on page 4-5
MDI containers are valid containers for applications displayed in traditional mode.	See "MDI Child application can be a traditional application" on page 4-6

Table 1. What's new in Informix Genero Desktop Client, v 2.40 (continued)

Overview	Reference
Traditional applications can now have a StartMenu to start sub applications.	See "'Pop-Tree' StartMenu support" on page 4-7
Debug console output can be configured to be more or less verbose. Items displayed are now categorized and you can decide which category is displayed in the console.	See "Debug console configuration" on page 4-8
The new --listen command line option and Active X setListeningMode() API function have been added to configure the tcp server.	See "Tcp server configuration" on page 4-8
The performance of the Genero Report Viewer has been improved.	See "Report Viewer: improved performances" on page 4-9

## Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM Software Support contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

Complete the following steps to contact IBM Software Support with a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM support topic in the *Software Support Handbook*.
2. Submit the problem to IBM Software Support in one of the following ways:
  - Online through the IBM Support Portal: You can open, update, and view all your Service Requests from the Service Request portlet on the Service Request page.
  - By phone: For the phone number to call in your region, see the Directory of worldwide contacts web page.



---

## Chapter 1. General

These topics introduce you to the Genero Desktop Client and provide guidance for starting and configuring this front-end.

- “What is the Genero Desktop Client”
- “Starting and Configuring the GDC”
- “Legal notices” on page 1-10

---

### What is the Genero Desktop Client

The **Genero Desktop Client (GDC)** is a graphical Front End for the Genero Runtime System. The Genero Desktop Client is multi-platform and can run under:

- Windows systems (32-bit and 64-bit packages available)
- Mac OS X (10.4, 10.5, 10.6). PowerPC® and Intel-based Macintosh are now supported. A Universal binary is provided.

**CAUTION:**

**Versions 10.3 and 10.2 are no longer supported with GDC 2.20.**

- Linux with X11 systems (32b and 64b packages available).

**Important:** libc 2.4 and libstdc++ 6 are now required.

The Genero Desktop Client can also be embedded in an HTML page using ActiveX. This solution works on Windows systems using the Internet Explorer browser.

---

### Starting and Configuring the GDC

These topics discuss starting and configuring the Genero Desktop Client.

#### Starting the GDC

To start the GDC:

- Under Windows systems, you can use the shortcut on the Start Menu.
- Under X11 systems, performing `envgdc` shell will add the Genero Desktop Client binary directory to your path; you will be able to start with the following command : `gdc`.
- Under OS X systems, the installer will add GDC to the Application folder.

By default, GDC will listen for Runtime System connections on port 6400. You can specify the port by starting GDC with the parameter `-p`.

If the port is not available, GDC will try the next port, continuing until it finds the first available one.

See command line for a list of all command line options.

#### Configuring the GDC

Click the Options icon to display the configuration options panel. The configuration options are organized across the following tabs: Preferences, Advanced, Connection, Security, and Report to Printer.

**Note:** Most of the options must be validated by pressing the "Apply" button. You can discard your changes by clicking the "Restore" button.

## The Preferences configuration panel

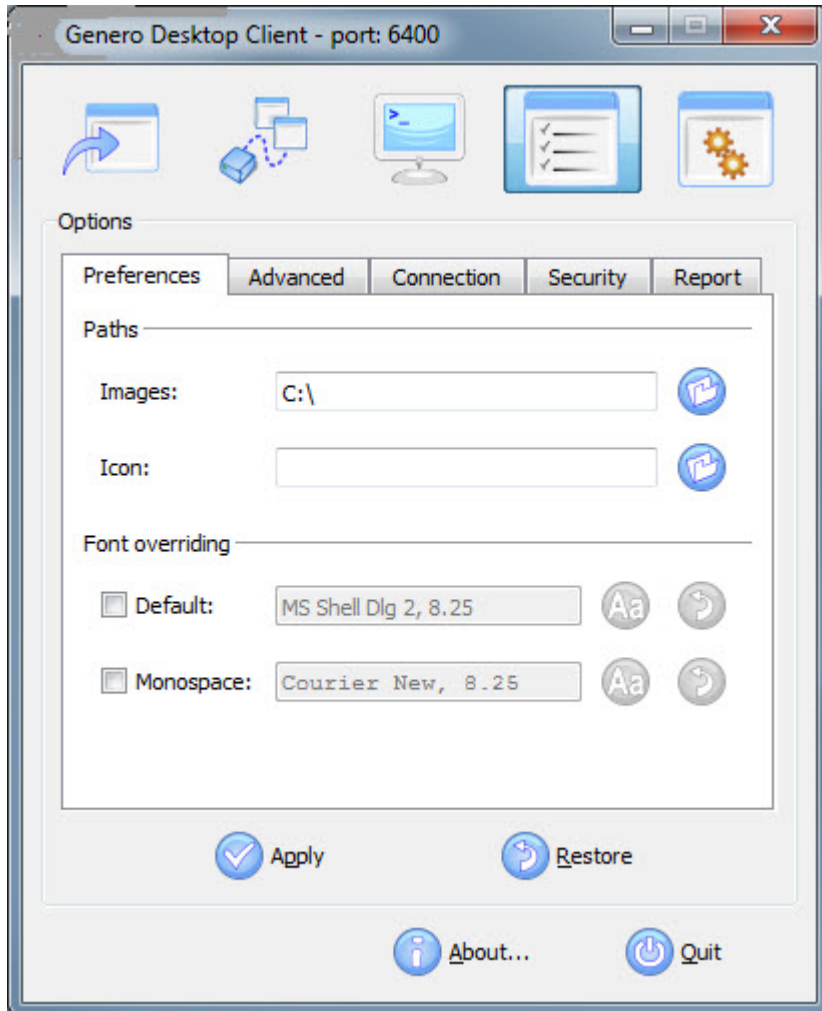


Figure 1-1. Genero Desktop Client Options; Preferences tab

The following options can be configured in the *preferences* panel:

- Path to local images
- Default Icon
- Default Font
- Fixed Font

**Path to local images:** Specifies the path for gdc to search when an image is needed. GDC will first check if the name provided corresponds to an absolute file name; then it will look in the path you have specified here, then in the /pics directory. If it still has not found the image, it will draw a "..." picture.

**Default Icon:** Specifies the default icon for GDC. This is the default icon used for the taskbar, the systray icon (under Windows systems), the shortcuts, the Terminals and applications.

**Default Font:** Specifies the default font for GDC. This font will be used everywhere in your applications.

**Fixed Font:** Specifies the default fixed font for GDC. This font will be used when the fixed font attribute is defined.

**Note:**

If you enter an invalid directory, the label turns red to warn you:



Figure 1-2. Images field with invalid directory entered

Most of the fields have auto completion:

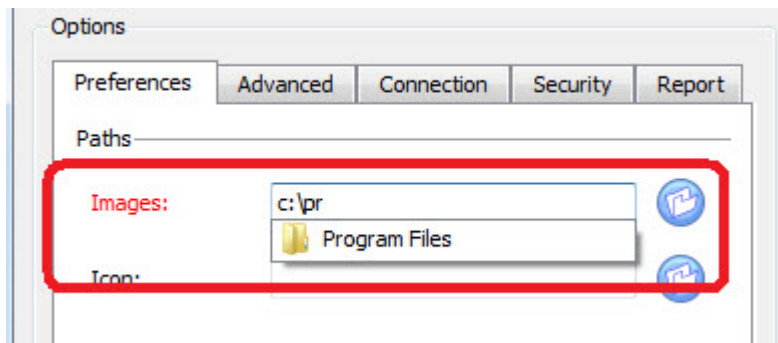


Figure 1-3. Images field with autocomplete feature

**Important:** Changes will not be applied until the "Apply" button is clicked.

## The Advanced configuration panel: Advanced options

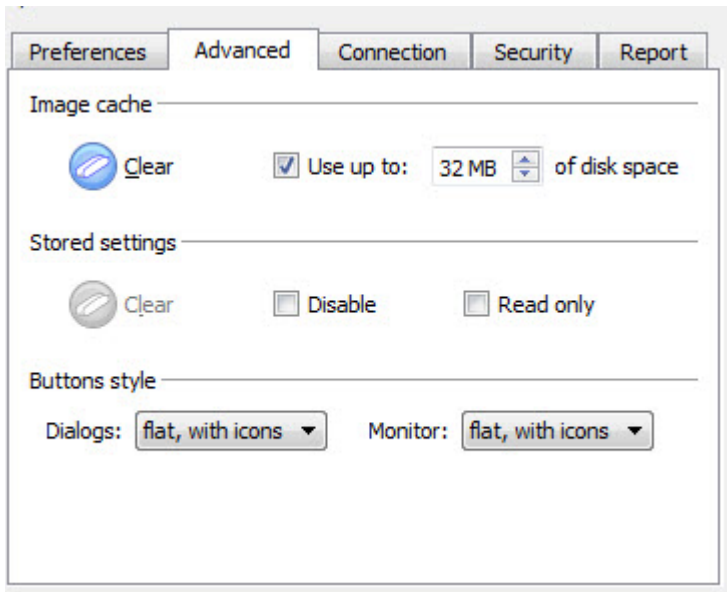


Figure 1-4. Genero Desktop Client Options; Advanced Tab

The following options can be configured in the *Advanced* panel:

- Disk Image Cache
- Stored settings
- Buttons style

### Image Cache

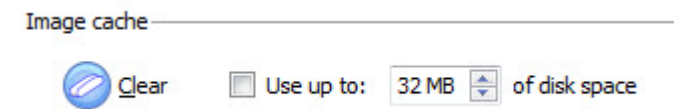


Figure 1-5. Image Cache section of Advanced Tab

GDC will locally store images which have been retrieved remotely. This can happen when the images have been found using http (either because the url specifies http, or a PICTURE alias is used with Genero Application Server), or on the Runtime System side (using FGLIMAGEPATH). The size of the image cache can be configured. Images are stored in %GDCDIR%\cache. When the cache is full, the images which haven't been recently used are removed from the cache. The Clear button will clear the cache.

**Tip:** As with previous versions, a memory cache is still used by GDC. Images that are frequently used are cached to be loaded as fast as possible. The Clear Cache button will clear both caches (memory and local disk).

## Stored settings

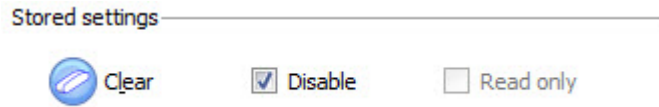


Figure 1-6. Stored settings section of Advanced Tab

Stored Settings can be temporarily disabled by checking "Disable". If "Read Only" is checked, GDC will read the stored settings when forms are loaded, but they won't be updated when forms are closed. If you want to clear settings, click on "Clear". this button is disabled if there are no stored settings.

**Attention:** We **strongly recommend** that you clear stored settings when migrating to a new main release of GDC (for instance, when moving from 2.1x to 2.2x). Otherwise, you might encounter some side effects due to corrections or new functionality.

## Buttons style

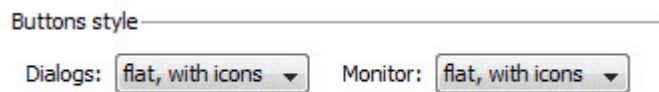


Figure 1-7. Buttons style section of Advanced Tab

The look of the monitor and dialogs (shortcuts wizard, login, about box, debug console) buttons can be customized to match the look-and-feel of a regular Genero application.

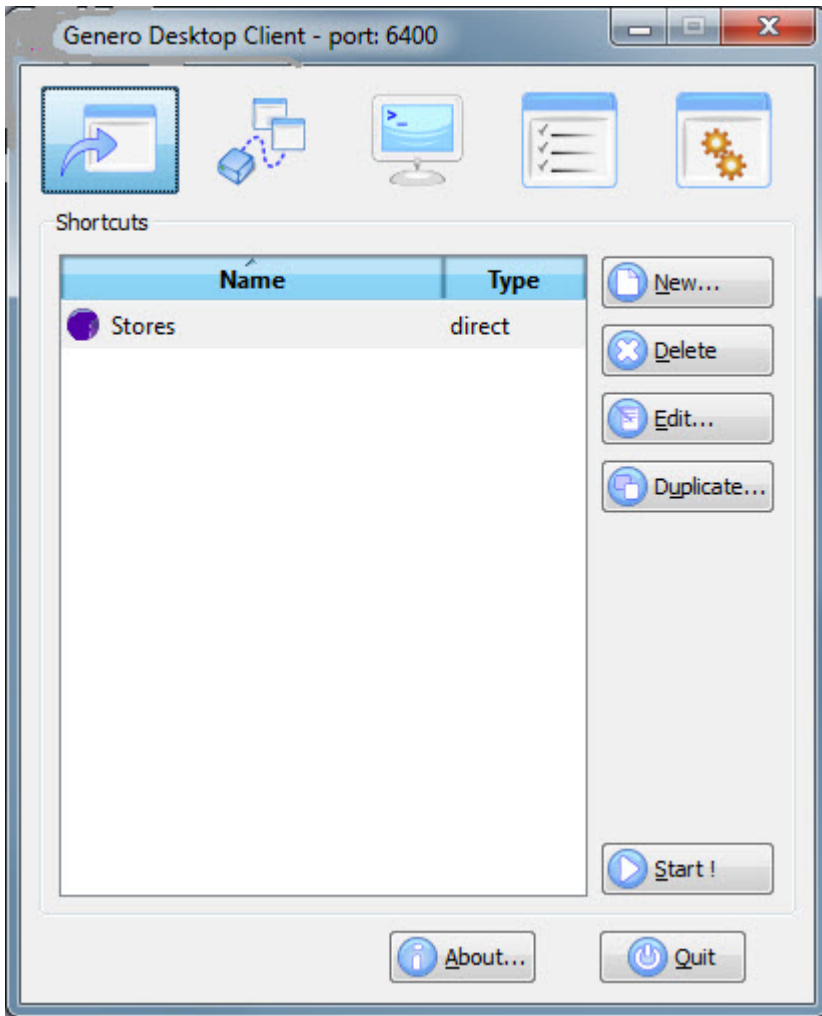


Figure 1-8. Raised buttons with icons

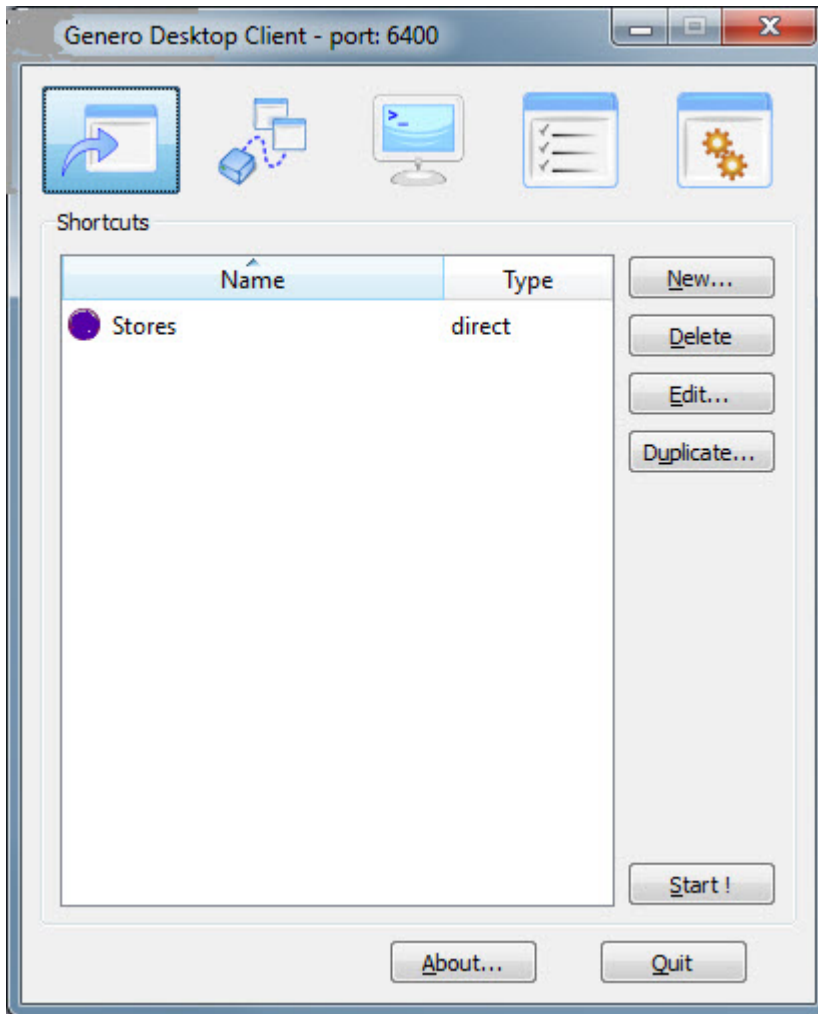


Figure 1-9. Raised buttons without icons

## The Connection configuration panel: Connection options

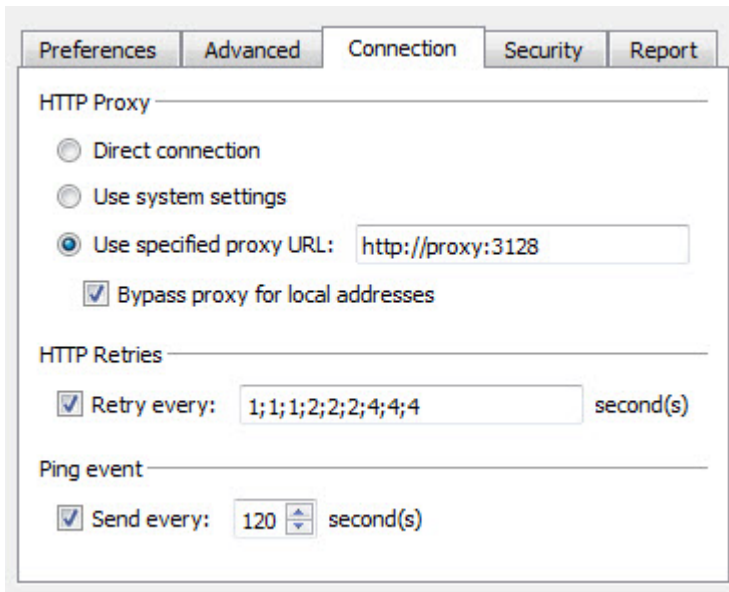


Figure 1-10. Genero Desktop Client Options; Connection Tab

These options can be configured in the *Connection* panel:

- Http Proxy
- Http Retries
- Ping event

### Http Proxy

In the HTTP Proxy section, you can set up the default proxy used for:

- Http shortcut (can be overridden in each shortcut)
- Http image lookup in Direct and Local shortcut

### Http Retries

In the HTTP Retries section, you configure how and when the GDC will resend the http request on socket or http error. If checked, the GDC will read the value from left to right, waiting the number of seconds entered between each separator before resending the failed request.

For example, the default value "1;1;1;2;2;2;4;4;4" means "on Socket/Http error, wait 1s before retrying, then, if the request still fail, wait 1s more, then 1s more, then 2s between each retry, then 4s between each retry".

Please note that this feature increases the time required for the detection of invalid hosts or dead servers, since the initial request will be retried at least 9 times with a total of 21 seconds to wait. You can temporarily disable it when creating a new shortcut, enabling you to quickly check the reachability of the server.



## Ping event

The purpose of a ping event is to check whether the connection with the runtime system or the application server is still alive. To perform this check, GDC sends a "ping" signal over the network. By default, the signal is sent every two minutes. The interval can be changed in the Ping event section (for instance, to 300 seconds).

## The Security configuration panel: Security options

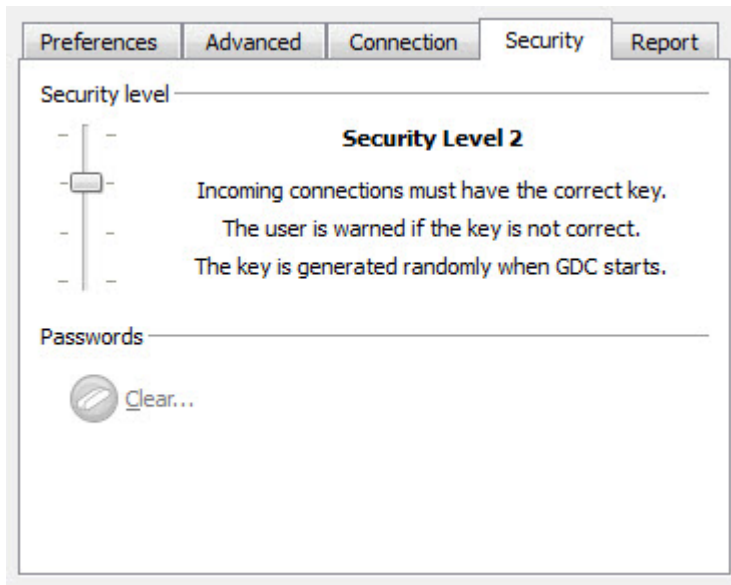


Figure 1-11. Genero Desktop Client Options; Security Tab

These options can be configured in the *Security* panel:

- Security Level
- Clear Password

### Security Level

The security level can be changed here. See Security for more details.

### Clear Password

Clears the passwords that are stored by GDC.

## The Report configuration panel: Report to Printer options

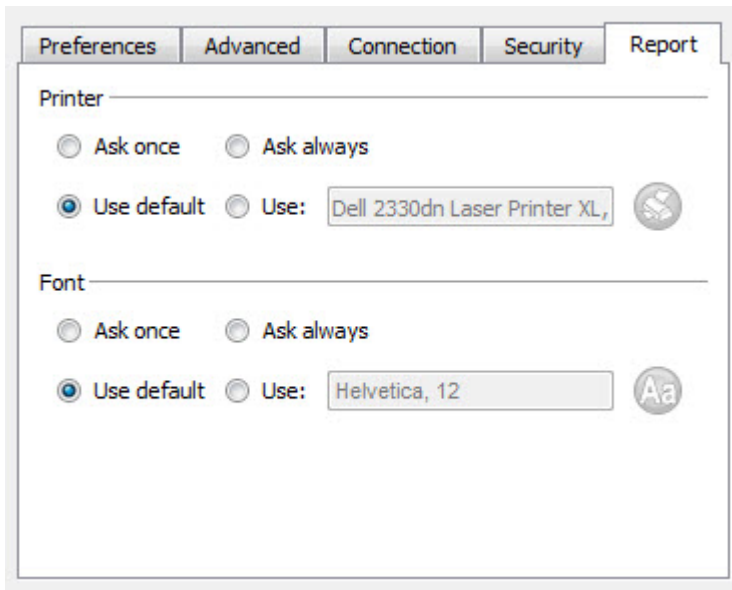


Figure 1-12. Genero Desktop Client Options; Report Tab

This panel provides some options for the font and printer used with REPORT TO PRINTER behavior:

- *ask once*: GDC will ask for the parameter once and then keep the choice in memory until it's closed.
- *ask on every report*: GDC will ask every time a report is printed.
- *Use default*: Use the system default printer or GDC's default font.
- *Use*: Use a specified printer or font.

### Applying a second configuration file

GDC will store most configuration options in config.xml file (located in %GDCDIR%\etc\config.xml). As of GDC 2.20, you can use an additional configuration file, via the --config command line option. Configuration options will then be read from:

1. the additional configuration file.
2. default config.xml if the option has not been defined in the additional config file.

If an additional configuration file is specified, changes will be stored in this file; the default config.xml file will not be altered.

---

## Legal notices

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes cryptographic software written by Eric Young ([eay@cryptsoft.com](mailto:eay@cryptsoft.com)).

This product includes software developed by CollabNet (<http://www.Collab.Net/>).

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed or owned by Caldera International, Inc



---

## Chapter 2. Applications

These topics introduce you to the applications side of Genero Desktop Client.

- “Shortcut System”
- “Shortcut Wizard” on page 2-4
- “Customizing your Login Box” on page 2-9
- “The Connections Panel” on page 2-11
- “The Terminals Panel” on page 2-13
- “The Debug Panel and Logging System” on page 2-14

---

### Shortcut System

The following topics discuss the shortcut system.

#### Shortcut System Overview

Genero Desktop Client (GDC) is able to store the information needed to start an application. The information is stored as a **shortcut**. You add a shortcut for each application. Shortcuts are stored the same way internally on each platform.

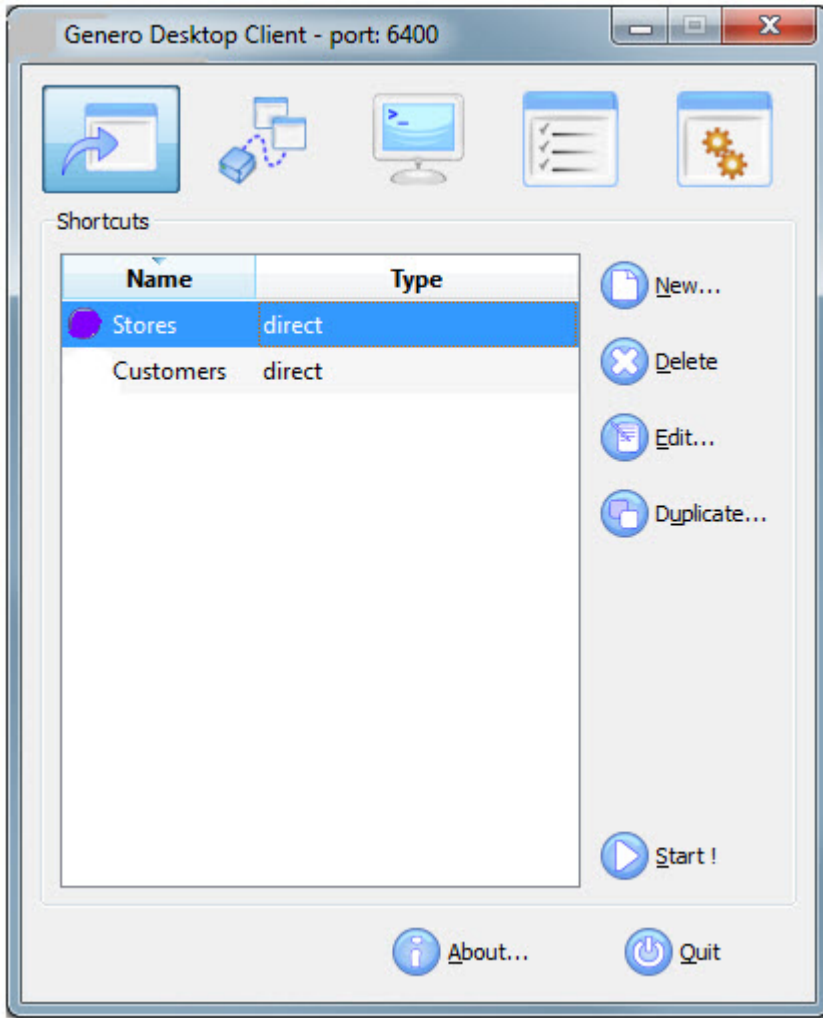


Figure 2-1. Genero Desktop Client User Interface (Administrator Mode)

## Administration Mode

By default, the Genero Desktop Client starts in *user mode*, where shortcuts and options cannot be modified. To create shortcuts or modify options, the Genero Desktop Client must be started in *admin mode* by using the "--admin" or "-a" command line option.

## Shortcut Management

Genero Desktop Client can interact with the Runtime System in different ways:

- The Runtime System is on a distant host and GDC will start it via telnet or SSH (Direct Connection),
- The Runtime System is on the same host and GDC will start it as a local application (Local Connection),
- The Runtime System is on a distant host, and GDC will be connect to it via Genero Application Server (Connection via AS).

## Create a new shortcut

Click on the "New..." button to start the Wizard to create a new shortcut.

## Edit shortcuts

Click on the "Edit..." button to start the Wizard to edit selected shortcut.

**Note:** If multiple shortcuts are selected, the edit wizard will be called for each selected shortcut. You can click on "Cancel" in the wizard to cancel the edition of the current and the following shortcuts.

## Duplicate shortcuts

Click on the "Duplicate..." button to create a copy of the selected shortcut.

If only one shortcut is selected, the Edit Wizard will open to allow you to modify the new copy.

If several shortcuts are selected, copies will be created with unique new names ; you'll be able to edit them afterwards.

## Remove shortcuts

Click on the "Delete" button to remove the selected shortcut.

## Import and export shortcuts

These options are available by right-clicking on the shortcuts list. Click on the "Import..." button to import a collection of shortcuts. Click on the "Export..." button to export a collection of shortcuts.

Shortcuts can be exported as a .gdc file. This xml gathers all configuration options describing one or more shortcuts. .gdc files can be used to transfer shortcuts between GDC installations. .gdc files can also be used to start a shortcut from the command line. See "Starting a Shortcut (Application)."

## Starting a Shortcut (Application)

The easiest way to start a shortcut (start an application) is to double-click the shortcut icon or by select the shortcut and pressing the "Start !" button:

Shortcuts can also be started via the command line:

- `gdc -S <shortcutname>` will start gdc (if needed) and the specified shortcut.
- `gdc myshortcut.gdc` will start gdc (if needed) and the shortcut defined in `myshortcut.gdc` file. If several shortcuts have been exported, the first one will be started.

## Local Shortcuts

By default, your shortcuts are saved into the `<GDCDIR>/etc/config.xml` file. You can also create **local** shortcuts that will be stored locally for each user. This feature can be useful if you share GDC on a network drive and don't want the user to modify the common shortcuts.

When config.xml file is read-only, any modification to a non-local shortcut will display a warning and create a local copy of the shortcut.

## Shortcut Wizard

These topics discuss the Genero Desktop Client shortcut wizard.

### Starting the Shortcut Wizard

When you start the Wizard, you will be asked for the name of the shortcut, and for an optional file name that will be used to display an icon associated with this shortcut. You may also store the shortcut in regedit as local for the current user.

You then must choose the Connection Type for your shortcut:

- **Directly to the DVM** - The Runtime System is on a different host, and GDC will start it via telnet, or SSH (Direct Connection)
- **By HTTP** - The Runtime System is on a different host, and GDC will connect to it via the Genero Application Server (Connection via AS)
- **Executing VM on local computer** - The Runtime System is on the same host, and GDC will start it as a local application (Local Connection).

### Direct Connection Shortcuts

In this mode, the Runtime System is directly connected to the GDC using TCP/IP network.

The first step is to specify :

- the hostname where the Runtime System is hosted. This can be omitted if you use the -host command line.
- the command line that will be executed to start the application on the Runtime System side.

Within the command line, you can use the following tags:

Table 2-1. Tags for use at the command line

Tag	Replaced by
@FGL	FGLSERVER=<IP Address>:<serv num> export FGLSERVER; FGLGUI=1; export FGLGUI

Table 2-2. You can use one of the @FGL variants depending on your system

Tag	Replaced by
@FGLNT	set FGLSERVER=<IP Address>:<serv num>&&set FGLGUI=1
@FGLCSH	setenv FGLSERVER "<IP Address>:<serv num>";setenv FGLGUI 1
@FGLKSH	FGLSERVER="<IP Address>:<serv num>";export FGLSERVER;FGLGUI=1;export FGLGUI
@SRVNUM	<GDC listening port - 6400 (The second part of FGLSERVER)>
@PORT	<GDC listening port>
@USR	<Client current user name>



Table 2-2. You can use one of the @FGL variants depending on your system (continued)

Tag	Replaced by
@LUSR	<Client current user name, lower case version>
@USER	<User name on the remote system>
@IP	<IP address of the client computer>
@COMPUTER	<Machine host name>
@E_SRV	export FGLSERVER
@4GLSRVVER	<GDC version>

These tags will automatically be replaced when the command is sent to the Runtime System host.

Additional environment variables may be set when using @FGL tags ; these variables are needed to manage connection checks.

To start your program on the Runtime System host, GDC will connect to the host using either telnet, SSH or SSH2. You can specify an alternative port, if your configuration needs it. Using SSH, tunneling can be established to secure your connection.

**Note:** This connection mode is only used when GDC connects to the host to start the application. Currently, the communication between the Runtime System and GDC does not use telnet, SSH, or SSH2.

If *Show Terminal Utility* is checked, the window of FGLTTY, our Emulation Terminal Utility, will be visible. (Please refer to the Terminals Section). This could help you check whether your command line is valid..

The *Backspace key sends Control-H* option modifies the sequence sent by the backspace key in FGLTTY. By default, **Control-?(127)** is used but you may change it to **Control-H**. This will allow you, for instance, to use the backspace key in dbaccess.

The *Start command in a new shell* option allows you to start a regular shell session before executing the remote host command. Note that this option is mandatory when using **automatic port forwarding**.

On the next screen, there is a method to **import your own login form**, using a .ui file (Qt designer's file format). See Customizing your own login form. You have the ability to force the login form to always be displayed on top, by checking the option **Always On Top**.

The same screen also allows you to select your **authentication method**:

- **Standard** : In this case, you have to provide the *username* you are using to connect to the host. This can be omitted if you use the -user command line.
  - If *Password required* is checked, GDC will ask you for a password. If your configuration allows you to connect without a password, uncheck this option. If a password is still requested, review your configuration.

**Important:** GDC will not modify your configuration to allow you to connect without a password. It is up to you or your administrator to manage this.

- If *Keep password* is checked, GDC keeps in memory the password you enter the first time you start a shortcut, and reuses it when you restart. The password is kept in memory, and is lost if you stop GDC.

**Important:** GDC never stores your password in a file or elsewhere unless *Allow persistent save* is checked. The password is kept in memory while GDC is launched, and is forgotten once GDC is stopped.

- If *and don't ask it again* is checked, GDC will only ask for the password the first time a shortcut is launched. After that, the password will be silently sent, without bothering the user.
- If *Even after restart* is checked, GDC keeps the password in memory even after having been stopped. Thus, the password is stored until it is manually cleared.

**Important:** GDC stores your password on disk in an encrypted form which is very difficult to read but not impossible. Someone with strong knowledge in cryptology can eventually break the password protection.

- **Kerberos** : On Windows platforms ( all versions after Windows 2000 ) you can also use Kerberos authentication if your user and computer are registered on an ActiveDirectory that provides a Kerberos interface. Using this authentication method, you are free to **Allow Ticket Forwarding**; this allows the SSH server to forward the Kerberos ticket that identifies the user to other processes. You may also select a **Server realm**; this identifies the Kerberos domain. This field can be mandatory, depending on the ActiveDirectory / Kerberos server configuration. Ask your System administrator for further details.
- **SSH private key file:** If you use an SSH connection, you can specify an ssh key file that contains the login information. The file format must use the PuTTY format and can be generated using PuTTY tools.

The next window allows you to specify **connection strings**. This table is used to tell GDC what to do when the Runtime System host displays a given string on the terminal. GDC can perform the following actions:

- Ask the user for a value, and send it back
- Display a message to the user
- Ask for a password
- Send the shortcut password
- Send the shortcut command
- Execute a local command and send the result
- Return a defined string
- Ignore the Runtime System string
- Send the login
- Get a free port number for Port Forwarding
- Show or hide the terminal
- Close the terminal

You can specify whether each string should be recognized only once or every time (check only once). Also, when **ignore remaining terminal strings** is checked, all the following strings are ignored.

The default terminal strings should be suitable in most of the cases, but you may have to adapt them to your system. For instance, the default string to send the command is `last login`: which may be different on your server.

Examples:

Table 2-3. Connection string examples

Recognized string	Description	Action performed by GDC
password:	This is the string used by the telnet daemon to ask for the password.	Sends the password
last login:	This is the string used by the telnet daemon to tell the user he has logged in successfully.	Sends the command
login:	This is the string displayed by the telnet daemon when the login has failed	Displays a message "Authentication has failed"

Please contact your System administrator if the default values are not appropriate.

The last screen let you configure fgltty options.

In 2.30.x, all these options are inherited from PuTTY. If you need more details on these options, please consult the PuTTY documentation.

## SSH Tunneling

Please refer to the GDC and SSH section for more information about SSH tunneling.

## Local Connection Shortcuts

In this mode, the Runtime System is on the same computer as GDC. To start your program on the Runtime System, GDC will simply start an executable (giving it some parameters). This executable will typically be:

- fglrun started in the application directory, or
- a batch file that will start all applications

You will have to provide the following information:

- the command line to select the executable
- the parameters needed by the executable
- the working directory

You can have, for example:

Table 2-4. Local connection shortcut examples

Command Line	Working Directory	Parameters	Remarks
fglrun	/home/fgl/demo/	stores.42m	fglrun should be in the PATH
c:\<mydir>\fgl\bin\fglrun.exe	c:\genero\demo	stores.42m	
c:\demos\stores.bat			stores.bat is a batch file that sets the environment and starts the program.

Table 2-4. Local connection shortcut examples (continued)

Command Line	Working Directory	Parameters	Remarks
C:\WINNT\system32\ CMD.EXE /C "d:\fjs\fgl\ envcomp.bat && fglrun D:\app\gift.42r"	D:\app		This will start envcomp.bat in FGLDIR to set the environment, then start the gift application.  The Working directory is the directory where fglrun can find the required files.

**Important:** If you have the configuration shown below, you must be sure that all the environment variables are set **before** starting the application. The variables can be set in the **Environment Variables** system dialog.

Table 2-5. Local connection shortcut example revisited

Command Line	Working Directory	Parameters	Remarks
c:\<mydir>\fgl\bin\ fglrun.exe	c:\genero\demo	stores.42m	

## Connections via the Genero Application Server

In this mode, the GDC will be connected to the Runtime System via the Genero Application Server, using the HTTP protocol.

The URL looks like: `http://myserver:6393/cgi-bin/fglccgi/ja/r/uideмо`

If your connection uses a proxy, you can configure it also. If you're attempting to connect to a local address, you can bypass the proxy.

You can also specify the Kerberos Realm on Microsoft Windows platforms if the web server requires a Kerberos authentication. This field is not mandatory for a Kerberos authentication, but it can be useful when many Kerberos servers are on the same network. Keep in mind that Kerberos authentication is designed to work in a local area network, and the results are unpredictable across a wide internet connection.

If the URL begins with https (secure), you can specify a client certificate to authenticate the client to the HTTPs server. For the moment, except for Microsoft Windows systems where you can use a system certificate, only two types of certificate are supported : PEM certificate which requires a certificate and a private key for this certificate, and PKCS12 certificate which includes both certificate and private key. If your certificate is password protected, you will be prompted for a password when the certificate is installed. Please note that the password may be requested again, depending on the state of the four checkboxes **Keep my password, and do not ask again, Allow persistent save and Even after restart.**

On Microsoft Windows, there are five methods of selecting a system certificate:

- SUBJECT: use the first certificate in which the subject field contains the given string.

- ISSUER: use the first certificate in which the issuer field contains the given string.
- HASH: use a hexadecimal hash that identifies a certificate. (eg: A5 C8 3F 34 21 C5 FF 8B 0A 0B 24 57 DD B2 C8 9F 1C 7A 45 76)
- ANY: select the first one
- ASK: ask the user to choose in a list

See the Genero Application Server documentation for more information on configuring applications.

## Shortcuts and environment variables

In some fields, GDC will replace any \$xxx (X11 / Mac OsX) or %xxx% (Windows) by the corresponding environment variables. The fields concerned are:

Table 2-6. Environment variable fields

Connection Type	Fields
direct	host, username, commandline
http	url
local	command line, working directory, parameters

If you want GDC to simply send the text instead of replacing the environment variable, use the "\" character to escape the variable (e.g. \\$HOSTNAME or \%HOSTNAME\%).

---

## Customizing your Login Box

### Topics

- Prerequisites
- How does it work?
- Tips

### Prerequisites

In order to build your own Login Box, you will firstly need to download Qt Creator. This software is available from the Qt website as stand-alone but Qt recommends to download it via the SDK. Download the version adapted to your Operating System.

### How does it work?

Default login box:

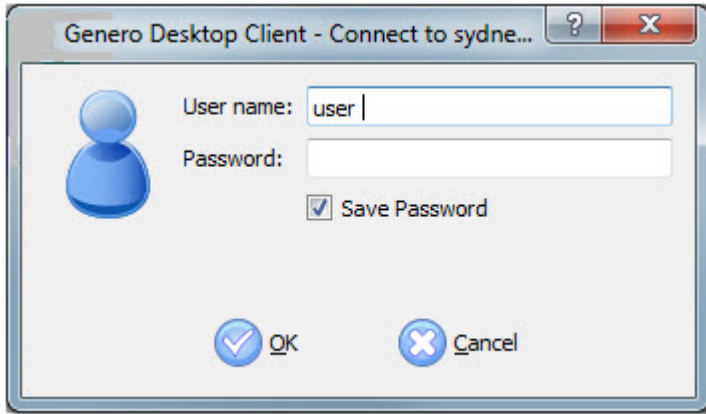


Figure 2-2. Default login box

To create the the login box you need to run the Qt Designer Form (included in Qt Creator). Select **New File or Project**, then in sub-category **File and Classes** select **Qt**, and, at the end, choose **Qt Designer Form**. Select **Widget** as template.

Now that you're in the Qt Designer Form, you have some rules to respect. You need the following items:

- a QWidget for the form
- a QPushButton named **m\_OKPushButton** for the OK button
- a QPushButton named **m\_CancelPushButton** for the cancel button
- a QLabel named **m\_UserNameLabel** for the label dedicated to the user name
- a QLabel named **m\_PasswordLabel** for the label dedicated to the password
- a QLineEdit named **m\_UserNameLineEdit** for the edit field where the user enters his name
- a QLineEdit named **m\_PasswordLineEdit** for the edit field where the user enters his password
- a QCheckBox named **m\_SaveCheckBox** for the checkbox which allows the password to be saved

Optional :

- a QLabel named **m\_TextLabel** if you're using a customized message when asking for the password again (See Connection Strings)

Then you can assign it to a shortcut and it will be used instead of the default login box.

## Tips

- As it is done in a Genero layout, use a Vertical Layout and Horizontal Layout as much as possible to correctly align and organize your widgets with each other.
- We strongly recommend you embed all elements in a Grid layout (QGridLayout). GDC will always resize the Login Box to its minimum size. When previewing (Alt+Shift+R or Tools -> Form Editor -> Preview ) your form in the Qt Designer Form, you should not be able to resize it to very tiny size. Using a grid layout around the various items should help to avoid this. The other solution is to specify a minimum size for the QWidget Form. For this, change the parameters (Width, Height) of the attribute **minimumSize** of the QWidget Form.

- Use Horizontal and Vertical Spacers to better control the free space.
- You are free to add some widgets which are usually not used in a login box: TextEdit, RadioButton, and so on.

---

## The Connections Panel

This topic provides information about the Connections panel.

### Overview

The "Connections" Panel lists applications that can be handled by GDC. For each application, it displays:

- **Name** - The name of the application. This refers to the text attribute of the UserInterface Node.
- **Id** - An internal identifier.
- **Type** - This refers to the way the application is connected: directly connected to the Runtime System (direct) or using HTTP protocol via Genero Application Server (http).
- **Date** - An indication when the application was started.

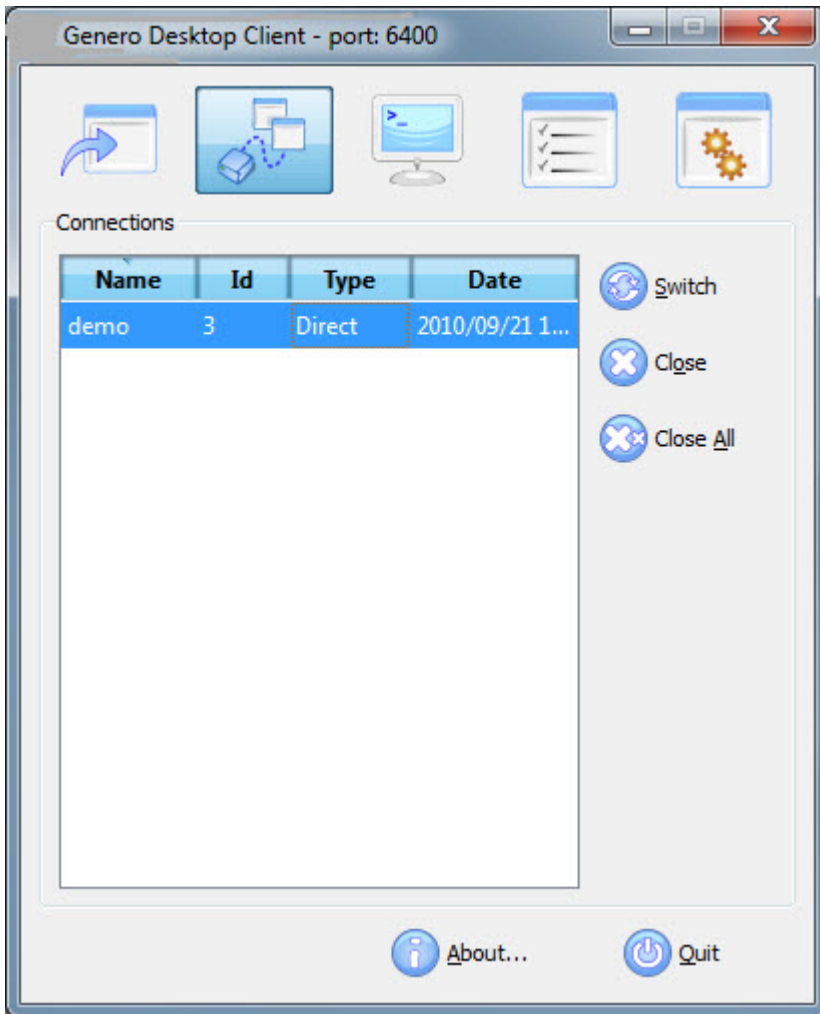


Figure 2-3. Connections panel

### Switch to / Close Buttons

If you click on the **Switch to** button, the selected application will be raised to the top, and the focus will be set on the current window. This will allow you to find your application easily if a lot of programs are launched.

If you want to stop applications, you can select them and click on the **Close** button. This will send the information to the Runtime System and the application will be stopped by GDC. **Close All** will close all running applications.

**Important:** GDC will first send a close request to the runtime system, which may be interpreted differently depending on your 4GL application settings (see **OPTIONS ON CLOSE APPLICATION** in the runtime system documentation), and will close the network connection after a given delay if the Runtime System does not react.



---

## The Terminals Panel

### Topics

- Overview
- Show/Hide
- Close / Close All

### Overview

Shortcuts use a terminal emulation utility (called `fglty`) to connect to the system hosting the Runtime System. Each line of the list in the Terminals panel refers to an active instance of the utility.

Terminals are automatically started by the Shortcut System.

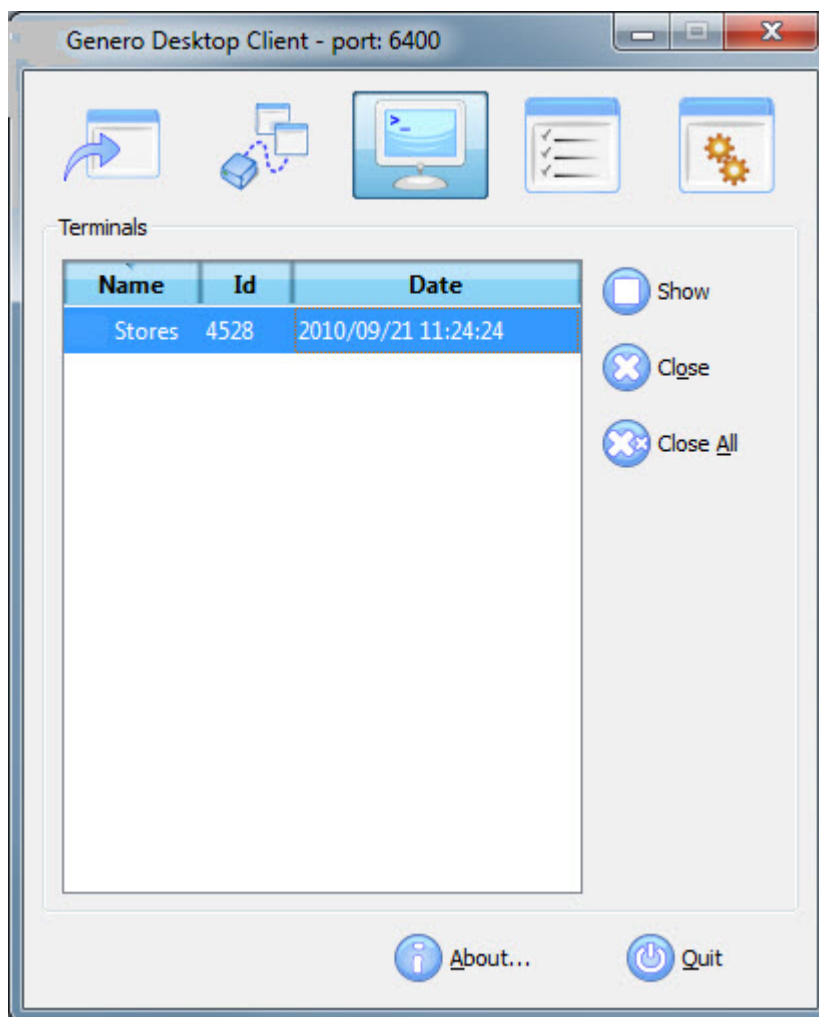


Figure 2-4. Genero Desktop Client; Terminals Panel

The terminal utility provided is called `fglty`.

**Important:** Coupled with GDC, its main purpose is to launch programs with the parameters which are set in shortcuts. You may use it as a strict terminal emulation utility, but we can not guarantee it will function well, and it won't be maintained for this purpose.

### **Show / Hide**

This button allows you to show or hide the selected Terminal. When you create a shortcut using the Shortcut Wizard, you can specify whether the Terminal Utility is shown. With this button you can show a hidden terminal, or hide a visible one.

This is typically used to check why your application has not started. Showing the Terminal Utility will display what has happened.

### **Close and Close All**

This button allows you to close selected Terminal Utilities. Close All will close all running Terminals.

**Important:** This may interrupt running applications as the Runtime System process may be terminated also.

---

## **The Debug Panel and Logging System**

### **Topics**

- Overview
- Logging system
- Debug Console

## Overview

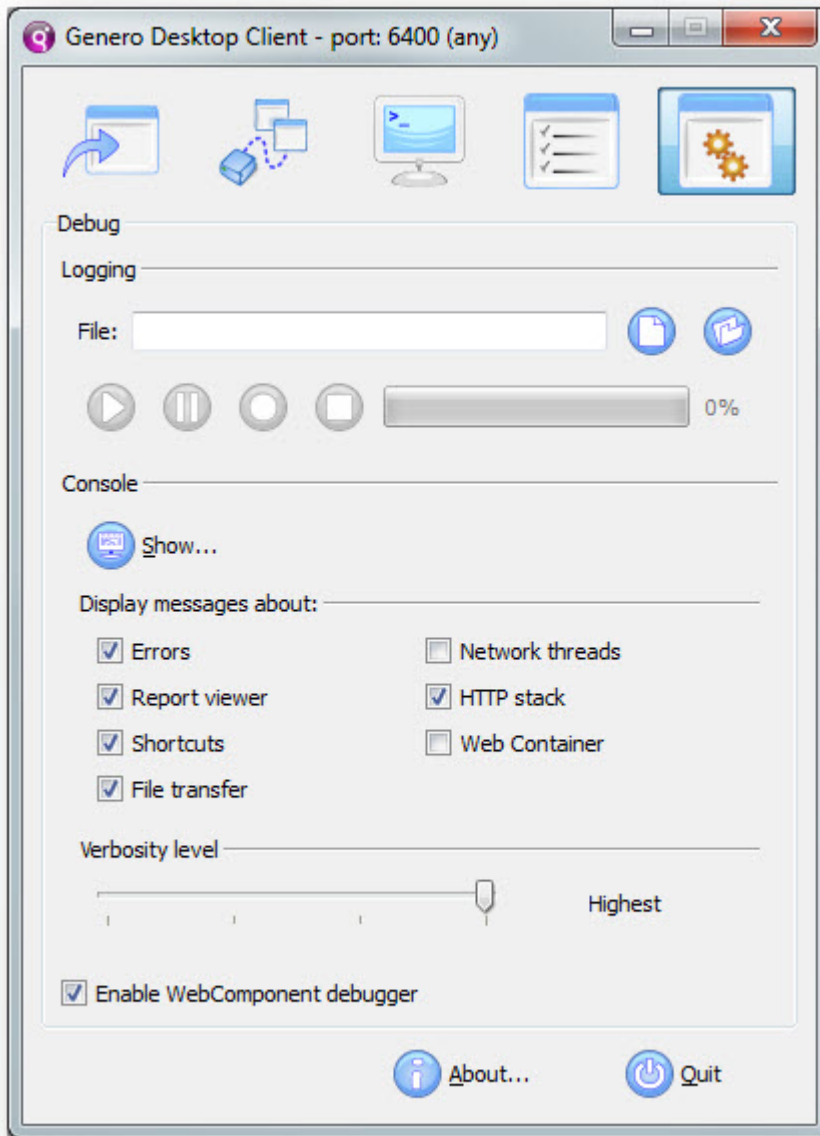


Figure 2-5. Debug panel

The Debug Panel shows the GDC debug facilities: the logging system and the debug console.

The Debug Panel is only available when GDC is started in debug mode.

### Using the Logging System

When GDC is started in debug mode, the logging system is available. This system will help you to:

- debug your applications
- create a demo

What is logged? The complete communication between the front end and Runtime System, so the Runtime System is not needed when you replay your demo.

**Important:** As only the communication is recorded, the "local-only" actions such as moving columns are not saved and replayed. Only the sent value of a field is saved; all user interactions (copy / paste, cursor, and so on) are not saved.

### **Recording Demo**

To record a demo, first select a log file to store the scenario. Then, click on the record button to start recording.

**Important:** Only applications you launched AFTER starting recording are stored.

### **Replay Demo**

To replay a demo, first select the log file where the scenario is stored. Then, click on the play button to start playing the demo. You can pause the replay by clicking on the pause button. The progress bar will indicate the progress of the demo.

**Important:** No user interaction is possible when replaying a demo, so you may have to stop recording the demo before the end of the application. Then, if you want to kill this application, you must use the Connections panel.

## **Viewing the Debug Console**

If you click on the **Console** button, a Window called Debug Console will appear.

**Important:** The debug console is only available in debug mode.

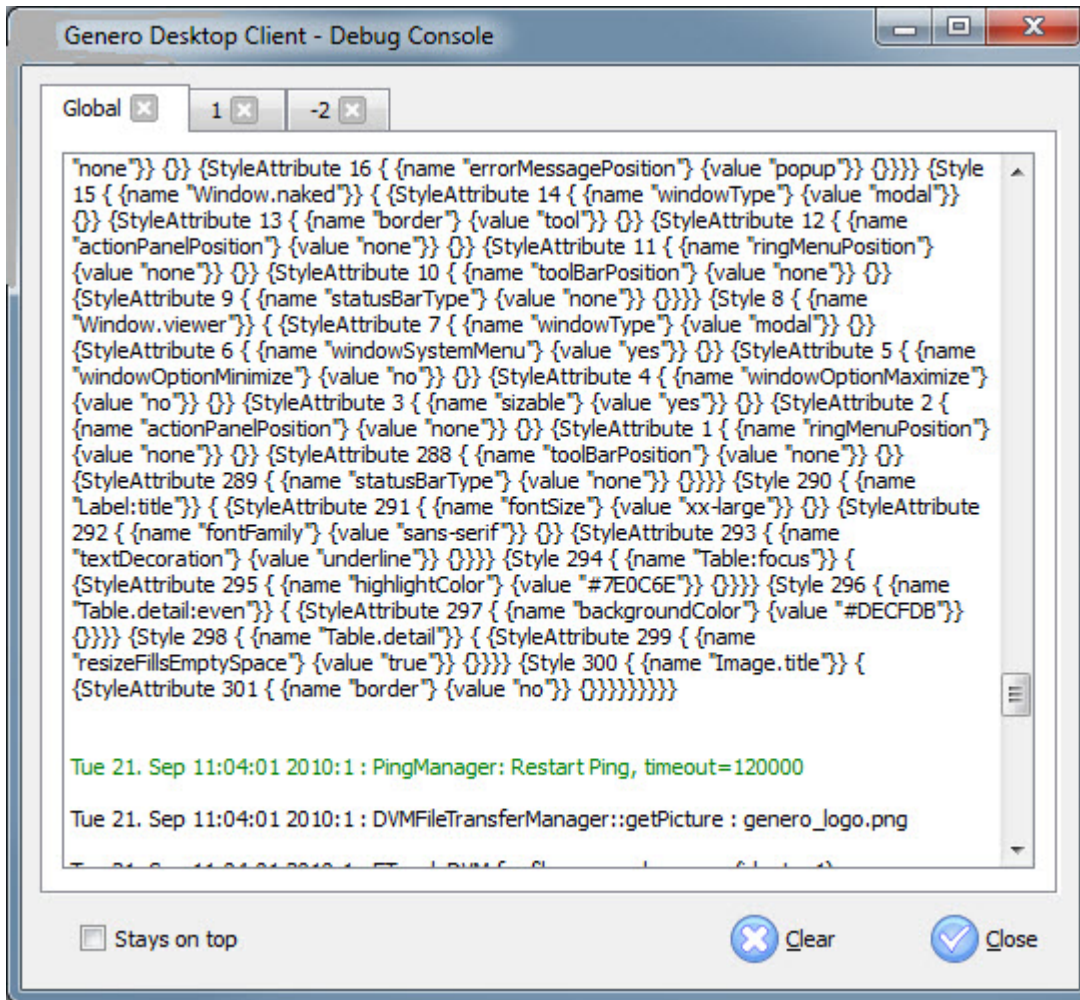


Figure 2-6. Debug console

In this window some debug information will be displayed. The information is categorized by color:

- in blue - what is sent by GDC to the Runtime System
- in black - what is received by GDC from the Runtime System
- in green - some comments or other information
- in red - error messages

This debug console could help you to see the communication between the GDC and the Runtime System. The first folder contains all communication threads which are also reported separately in the other folder tabs (one for each application).

If you want the debug console to stay in foreground and have it always visible, check the Stays On top check box.



---

## Chapter 3. Features

These topics introduce you to the features of the Genero Desktop Client.

- “The Command Line”
- “Printing a screen shot” on page 3-5
- “Local actions” on page 3-6
- “Localization encoding list” on page 3-8

---

### The Command Line

These topics discuss using the command line with the Genero Desktop Client.

#### Command line options

GDC handles the following options:

Table 3-1. GDC command line options: Information

Option	Parameter	Description
-h		Displays About Box and exits.
-V		
--Version		
-c		Defines an additional configuration file. See Applying a second configuration file.
--config		

Table 3-2. GDC command line options: Network, System

Option	Parameter	Description
-n		Starts a new instance of GDC.
--new		
-p	new_port	GDC will listen on the first available port starting with <i>new_port</i> . <b>Important:</b> If an instance is already running, -p has no effect if -n is not specified.
--port		
-q		If the expected port (either 6400, or port specified by --port) is not available, GDC will stop (exit with -1).
-D		Starts GDC in debug Mode (debug Tree and debug Console are active)
-A	security level	Sets GDC's security level regarding the Runtime System's connection.
--Authentication		

Table 3-2. GDC command line options: Network, System (continued)

Option	Parameter	Description
--listen	ANY LOCALHOST NONE AUTO	<p>Specify the network listening mode of the GDC.</p> <p>ANY: Listen to any network for a new connection (old behavior)</p> <p>LOCALHOST: Listen to localhost only, DVM must be on the same host as GDC or must be on a host connected with port forwarding</p> <p>NONE: No listening at all. Only HTTP connection will work, this is the most secure operating mode.</p> <p>AUTO: Like LOCALHOST, but GDC will switch back to ANY when a regular direct shortcut (without port forwarding) is used, to allow a connection from outside. The GDC will switch back to LOCALHOST when no more connections and no more terminals are active, after a two minutes timeout.</p> <p><b>Important:</b> AUTO is the default, which means connections from the outside that are launched without using the shortcut system will not work anymore.</p>

Table 3-3. GDC command line options: Start Application

Option	Parameter	Description
	.gdc file	Starts GDC with the shortcut specified in the .gdc file. If the file contains several shortcuts, it starts with the first alphabetically. See Import and export shortcuts.
-S	shortcut_name	If GDC has not been launched, GDC will start minimized; then, the shortcut named shortcut_name will be started.
--Start		
-s		If GDC has not been launched, GDC will start minimized, using the information given by -U, -H, -T, -P and -C to connect to a DVM.
--startDirect		



Table 3-3. GDC command line options: Start Application (continued)

Option	Parameter	Description
-U	user name used	The specified user name will be used when a Direct Connection starts.
--User		This option can be used if you share GDC; then each user can create a link to the bin and differentiate the shortcut that will be launched.
-H	host name	The specified host name will be used when a Direct Connection starts with a defined shortcut (with -S), or starts directly (with -s).
--Host		
-P	password	The specified password will be used when a Direct Connection starts with a defined shortcut (with -S), or starts directly (with -s).
--Password		
-K		The password specified with -P option will be kept in memory and no longer requested.
--KeepPassword		
-C	command_line	The specified command_line will be used when a Direct Connection starts with a defined shortcut (with -S) or starts directly (with -s).
--Cmd		
-T	connexion_type	Defines which protocol should be used when an application starts with -s. Values can be: TELNET, SSH, SSH2. Default is SSH2. <b>Note:</b> Default is now SSH2.
--Type		
-w		Defines whether the terminal window is visible (when --startDirect option is used). The terminal is hidden by default.
--ShowTerminal		
-f		If a password is provided with --Password, GDC won't display a login box when starting a shortcut. If you explicitly want the login box to be shown, with password and user pre-entered, use the -f option.
--ShowFirstLogin		
-e		Allows the user to save the password in a persistent way (It will not be asked again, even if GDC is stopped and restarted).
--AllowPersistentSave		

Table 3-3. GDC command line options: Start Application (continued)

Option	Parameter	Description
-k	Putty Key file (.ppk)	Uses the given Putty Key File as authentication method when Direct Connection.
--PuttyKey		
-u	Genero application URL	Starts the HTTP Genero application given by the URL.
--url		
-g	remote .gdc file	Starts directly the remote .gdc file.
--gdcfile		

Table 3-4. GDC command line options: Start GDC

Option	Parameter	Description
-a		Starts GDC in admin mode.
--admin		
-M		Starts GDC minimized.
--Minimized		
-X		Closes GDC if there is no longer an application or terminal running.
--close		
-i		GDC starts with <b>ignore Stored Settings</b> on.
--ignoreSettings		

Table 3-5. GDC command line options: Logging Mechanism

Option	Parameter	Description
-l	Log file	Starts GDC and replays the given Log File
--logplay		
-L	Log file directory	Starts GDC and replays all the Log Files inside a givendirectory
--logdir		
-r	Log file	Starts GDC, records a log, and saves the given Log File.
--logrec		
-t	delay	By default, replays Log Files at their recording speed. You can change the delay (in milliseconds) between the steps.  <b>CAUTION:</b> <b>A delay that is too small will overcharge GDC. Please consider 100 milliseconds as the smallest acceptable value.</b>
--logtimeout		

## Mac OS X users

The command line can be used in either of the following ways:

- Starting the terminal application (Applications, Utilities), and then:  
./Applications/gdc.app/Contents/MacOS/gdc <command line>  
. OSX expects the path to be absolute and not relative.
- Using the following Apple Script:  
do shell script ".Applications/gdc.app/Contents/MacOS/gdc  
<command line>"

## Warnings

- The `-S` and `-s` options must be used separately; `-S` is used to start an existing shortcut, and `-s` to start an application using the command line.
- When using `-s`, you must specify at least the host and the command line. The username and password will be prompted if needed.
- Even if you're using the `-q` option, GDC will first check whether another instance is already running. If you really want your GDC instance to stop if the port is not available, use `-n` and `-q` together. Using `-q` alone will stop GDC if the port is not free and not being used by another GDC.

## Command line examples

- `gdc -p 6350`  
Starts GDC on port 6350.
- `gdc -S demo`  
Starts GDC, and the shortcut named **demo**.
- `gdc -S demo -U smith`  
Starts GDC, and the shortcut named **demo** using **smith** as the user name.
- `gdc -s -T SSH2 -U smith -H server -P whatisthematrix -C "cd demo ; fgln demo" -X`  
Starts GDC, then connects to **server** as the user **smith** with the password **whatisthematrix**. Once connected, performs the specified command line **cd demo ; fgln demo** and closes the GDC when all the applications or terminals are over.

---

## Printing a screen shot

GDC provides a feature to send the current window to any installed printer. This will allow you to print a screenshot directly, without any other tool.

To call this feature, you can:

- press CTRL + ALT + P
- press ALT + Print Screen (under Linux systems only, under Windows this combination will be used by the system to put the current screenshot into the clipboard)
- Select the "Hardcopy" option in the System Menu (Windows only)

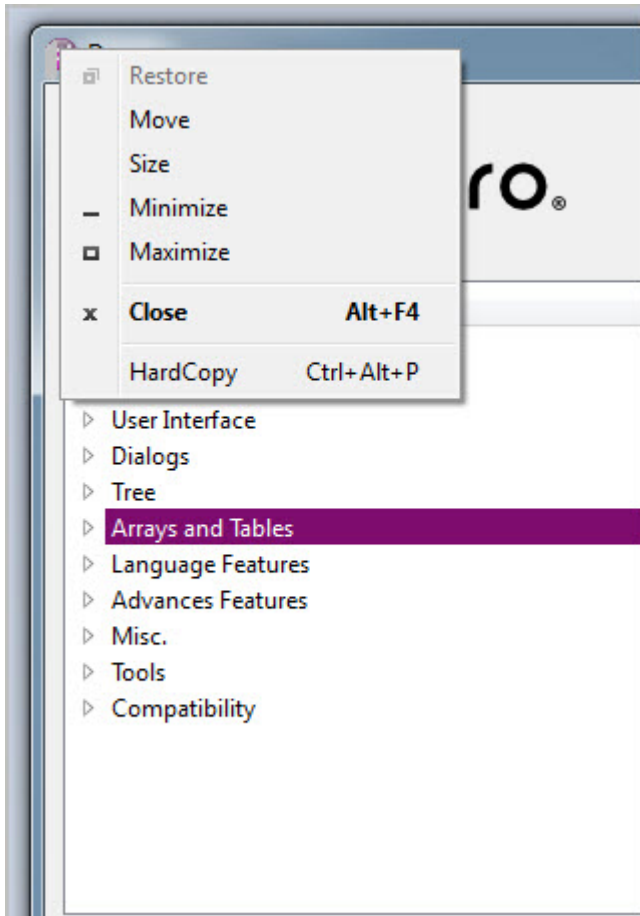


Figure 3-1. Print contextual menu

The classic "Print dialog" opens, allowing you to select the desired printer, configure it, and then print the current window.

---

## Local actions

### Local actions overview

Some features of the GDC are defined as "local", including "completely local" features like copy, cut or paste. Some others depend on the DVM but concern local behavior, like the navigation in a table.

To allow you to customize these features with an accelerator, images, comments, and so on, the features are integrated as **local actions**. They follow the same rules as Runtime System actions, but they are created by the Front End instead of the Runtime System.

You will be able to create actionViews for these actions, as you can for any other action.

#### Example:

```
01 BUTTON btn1 = nextfield;
```

When this button is pressed, the focus will go to the next field.

**Example:**

```
01 <ActionDefault name="nextfield" accelerator="return">
```

The "return" key will be the accelerator to go to the next field.

## List of local actions

Topics:

- Editing
- Navigation in fields
- Navigation in tables
- Selection in tables
- Navigation in folder page
- Interrupt

*Table 3-6. Local Actions: Editing*

Local Action	Description	Shortcut Hotkeys
editcopy	copies the selected text into the clipboard	CTRL+C
editcut	cuts the selected text into the clipboard	CTRL+X
editpaste	pastes the content of the clipboard into the current field	CTRL+V

*Table 3-7. Local Actions: Navigation in fields*

Local Action	Description	Shortcut Hotkeys
nextfield	goes to the next field	TAB
prevfield	goes to the previous field	SHIFT+TAB

*Table 3-8. Local Actions: Navigation in tables*

Local Action	Description	Shortcut Hotkeys
firstrow	goes to the first row	HOME
prevpag	goes back one page (with TABLE, it follows Internet Explorer behavior)	PRIOR
prevrow	goes to the previous row	KEY_UP
nextrow	goes to the next row	KEY_DOWN
nextpag	goes forward one page (with TABLE, it follows Internet Explorer behavior)	NEXT
lastrow	goes to the last row.	END

*Table 3-9. Local Actions: Selection in tables*

Local Action	Description	Shortcut Hotkeys
selectall	Select all rows (if MultiSelection is enabled)	CTRL+A

Table 3-10. Local Actions: Navigation in folder pages

Local Action	Description	Shortcut Hotkeys
prevtab	raises the previous page	CTRL+SHIFT+TAB
nexttab	raises the next page	CTRL+TAB

Table 3-11. Local Actions: Interrupt

Local Action	Description	Shortcut Hotkeys
interrupt	sends interrupt to the Runtime System	

Other local actions are those related to the Rich Text Editing.

---

## Localization encoding list

Localization Support: The following encodings are supported by GDC.

Table 3-12. Localization encoding list

Encoding List
Apple Roman
Big5
Big5-HKSCS
EUC-JP
EUC-KR
GB18030-0
IBM 850
IBM 866
IBM 874
ISO 2022-JP
ISO 8859-1 to 10
ISO 8859-13 to 16
Iscii-Bng, Dev, Gjr, Knd, Mlm, Ori, Pnj, Tlg, and Tml
JIS X 0201
JIS X 0208
KOI8-R
KOI8-U
MuleLao-1
ROMAN8
Shift-JIS
TIS-620
TSCII
UTF-8
UTF-16

Table 3-12. Localization encoding list (continued)

Encoding List
UTF-16BE
UTF-16LE
UTF-32
UTF-32BE
UTF-32LE
Windows-1250 to 1258
WINSAMI2





---

## Chapter 4. Upgrading and New Features

These topics introduce you to new features of each release of Genero Desktop Client and provide guidance for upgrading to the next version.

- “Genero Desktop Client 2.40 New Features”
- “Genero Desktop Client 2.32 New Features” on page 4-9
- “Genero Desktop Client 2.30 New Features” on page 4-9
- “Genero Desktop Client 2.22 New Features” on page 4-18
- “Genero Desktop Client 2.21 New Features” on page 4-20
- “Genero Desktop Client 2.20 New Features” on page 4-26
- “Genero Desktop Client 2.2x migration guide” on page 4-38
- “Genero Desktop Client 2.3x migration guide” on page 4-43
- “Genero Desktop Client 2.4x migration guide” on page 4-45

---

### Genero Desktop Client 2.40 New Features

This section describes the new features for Genero Desktop Client 2.40

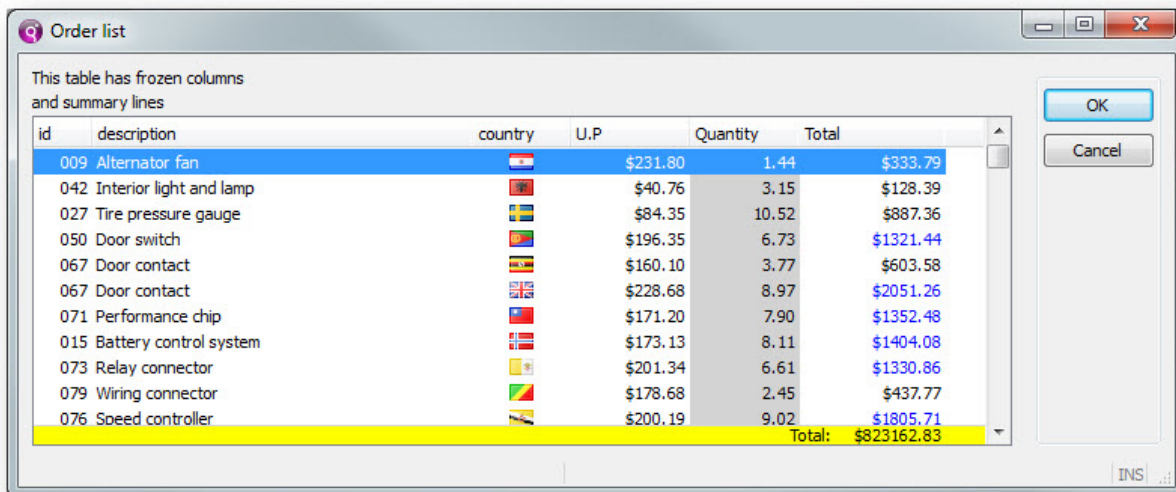
These topics organize the new features of the 2.40 release of Genero Desktop Client by categories.

#### Genero Features: Genero Desktop Client 2.40 New Features

This section introduces the Genero features-related new features in Genero Desktop Client 2.40.

##### Summary Line Support

Support for Summary Line introduced with Genero BDL 2.40.



This table has frozen columns and summary lines

id	description	country	U.P	Quantity	Total
009	Alternator fan		\$231.80	1.44	\$333.79
042	Interior light and lamp		\$40.76	3.15	\$128.39
027	Tire pressure gauge		\$84.35	10.52	\$887.36
050	Door switch		\$196.35	6.73	\$1321.44
067	Door contact		\$160.10	3.77	\$603.58
067	Door contact		\$228.68	8.97	\$2051.26
071	Performance chip		\$171.20	7.90	\$1352.48
015	Battery control system		\$173.13	8.11	\$1404.08
073	Relay connector		\$201.34	6.61	\$1330.86
079	Wiring connector		\$178.68	2.45	\$437.77
076	Speed controller		\$200.19	9.02	\$1805.71
Total:					\$823162.83

Figure 4-1. Summary line

## Built-in search and fast seek

Support for built-in search and fast-seek features introduced with Genero BDL 2.40.

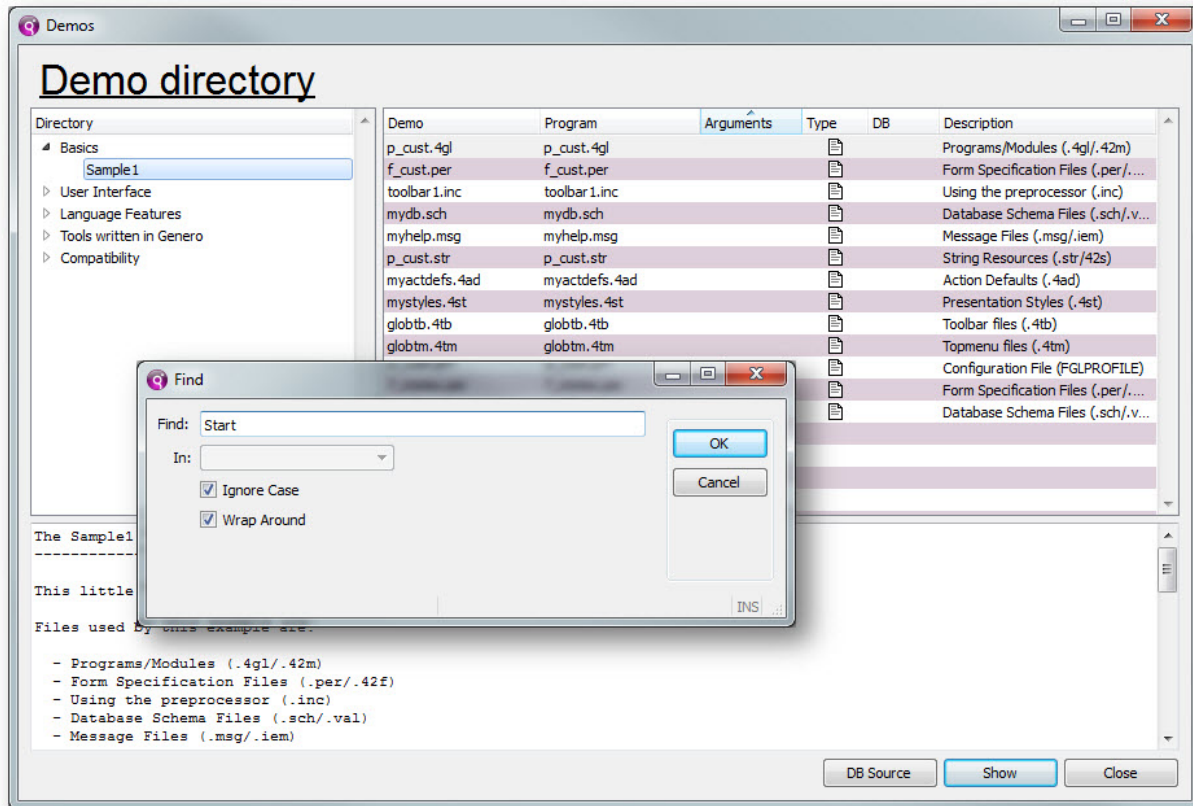


Figure 4-2. Built-in Search

## Widgets: Genero Desktop Client 2.40 New Features

This section describes the widget-related new features in Genero Desktop Client 2.40

### ComboBox autocompletion timer

Combobox items can be selected by pressing the first letters within a short period of time. After this delay, the lookup restarts.

For instance, pressing "par" will select "paris", but "pa (pause) r" will select "paris" and then "roma". The delay is by default 400 ms, and the new style attribute `completionTimeout` allows to configure it.

This style attribute also applies to RadioGroups.

### ComboBox completer

This topic describes the behavior of the style `comboboxCompleter`.

When the style `comboboxCompleter` is active, the ComboBox will have the following behavior:

- The combobox is editable, but only characters that match an item in the list are allowed (if the list contains the item "aa" and the item "ab", you can type "a", "aa", "ab", but nothing else. If you paste text in the field, it will be truncated until the rule is fulfilled).
- The dropdown list will only display item which starts with the same characters as the edit field. It is dynamically updated as you type (if the list contains the item "aa" and the item "ab" and you type "a", you will see both item displayed, but if you continue to type another "a", you will only see "aa" in the list).
- The best match is automatically selected when leaving the field (thus performing an "on change") as soon as you hit "TAB" key, even if the input is not complete.

### Table line decoration review

Style decoration that are applied on a given line (using :odd/:even pseudo selectors for instance) are now applied on the whole line, including the right hand side area where there may be no column.

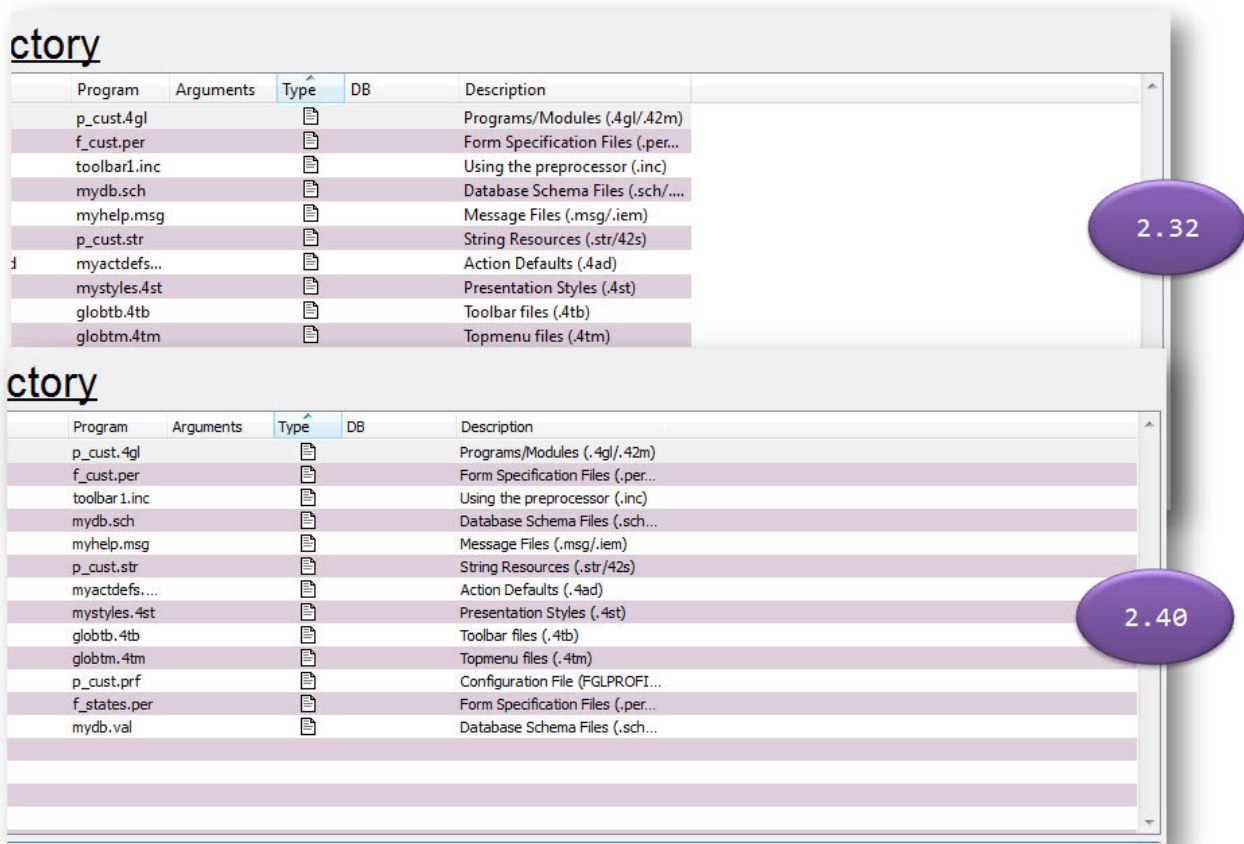


Figure 4-3. Table decoration spans the whole line

### Web Component: Debugging information

This topic introduces the debugging information added to assist with debugging the WebComponent.

Introduced in 2.30, WebComponent is a powerful mechanism that allows you to integrate any "web based" component in your 4GL application. Because of the fully

integrated aspect, it was sometimes difficult to setup web components within GDC, and to understand what could go wrong (for example: JavaScript error). GDC 2.40 is now showing new debugging information in the debug console:

- JavaScript Messages
- WebComponent internal debugging info: url loaded, http errors
- gICAPI object debugging info: creation, bridge (GDC // JavaScript) setup
- API debugging: calls to onData, onProperty, onFocus, setData, setFocus, Action

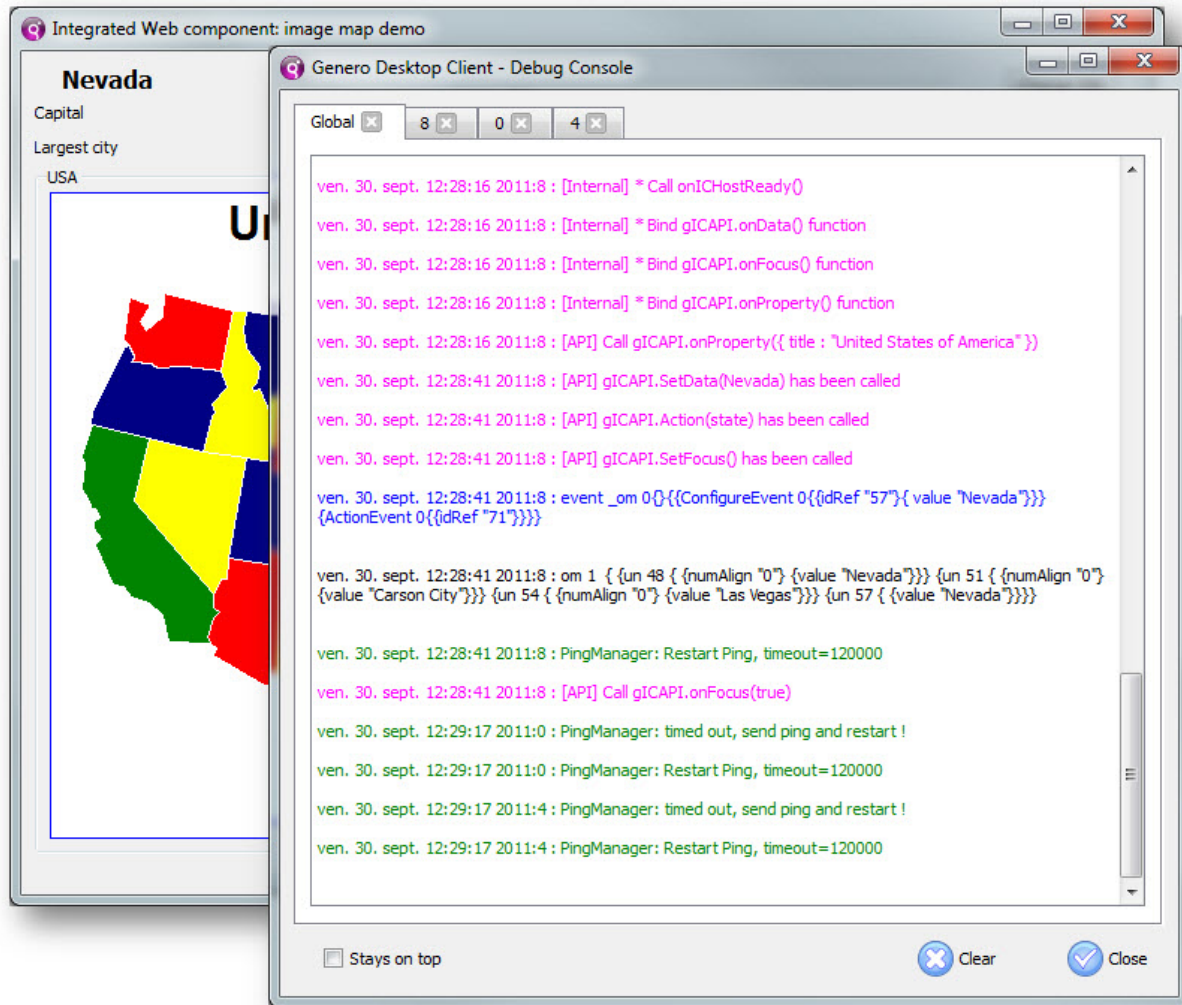


Figure 4-4. Debug console

You will have to enable webcomponent debugging in the console to see the messages.

### Web Component: gzip support

WebComponent now accepts gzip encoding.

### Web Component: cookies support

WebComponent now accepts cookies, which could be useful for authentication purposes.

Cookies are kept in memory during the lifetime of the GDC.

## **Traditional Mode: Genero Desktop Client 2.40 New Features**

This section describes the traditional-mode related new features for Genero Desktop Client 2.40

### **Global toolbar is now supported**

Global (from CALL ui . interface.LoadToolBar()) Toolbar is now displayed in Traditional mode.

Form toolbars are still not displayed, as it makes no sense in the traditional mode context.

### **Status bar support**

Traditional windows can be configured to have a status bar.

COMMENT, ERROR and MESSAGES will be displayed there instead of their own LINE.

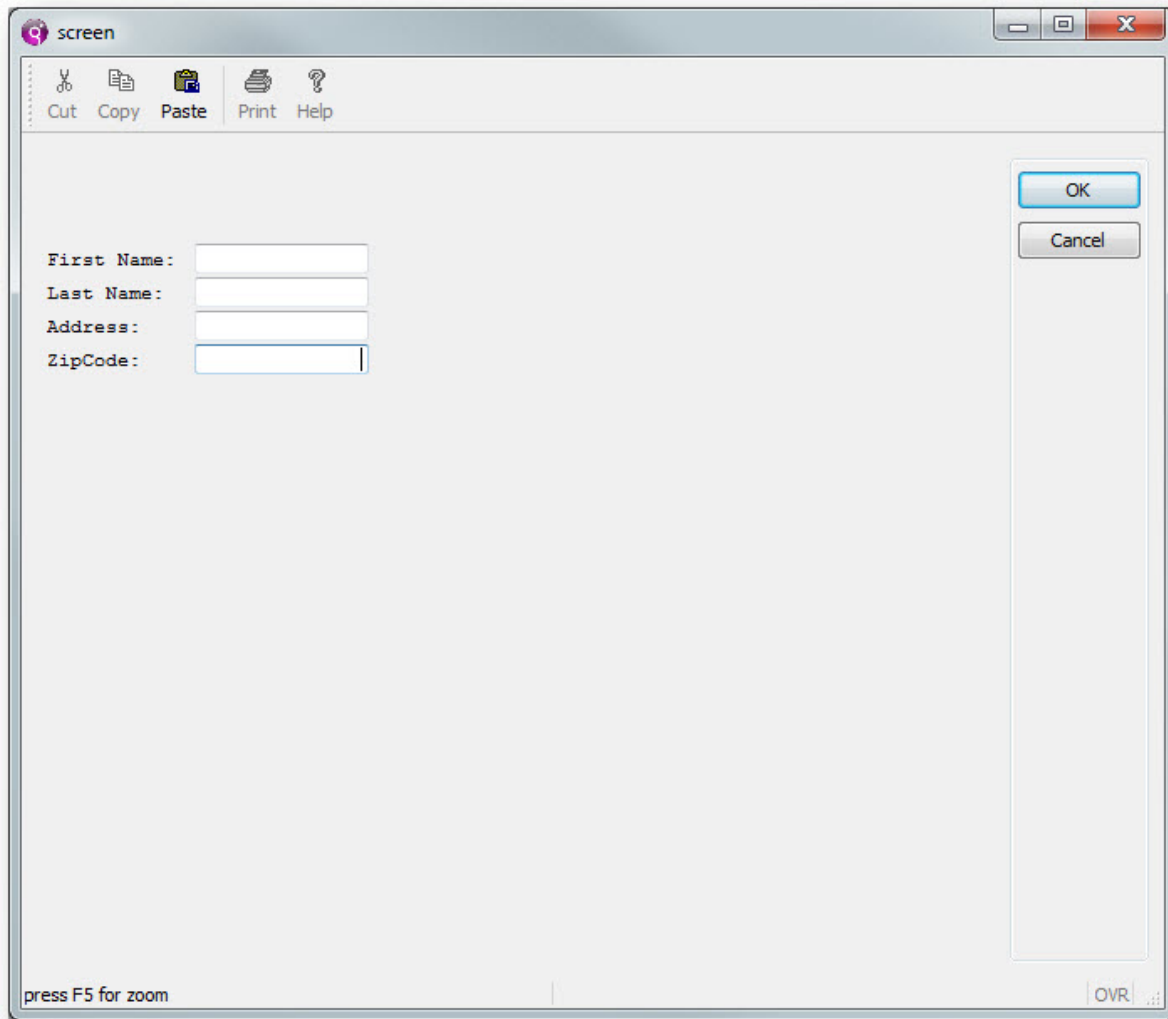


Figure 4-5. Status bar

### **MDI Child application can be a traditional application**

MDI Container can now be container for traditional applications (and "modern" at the same time).

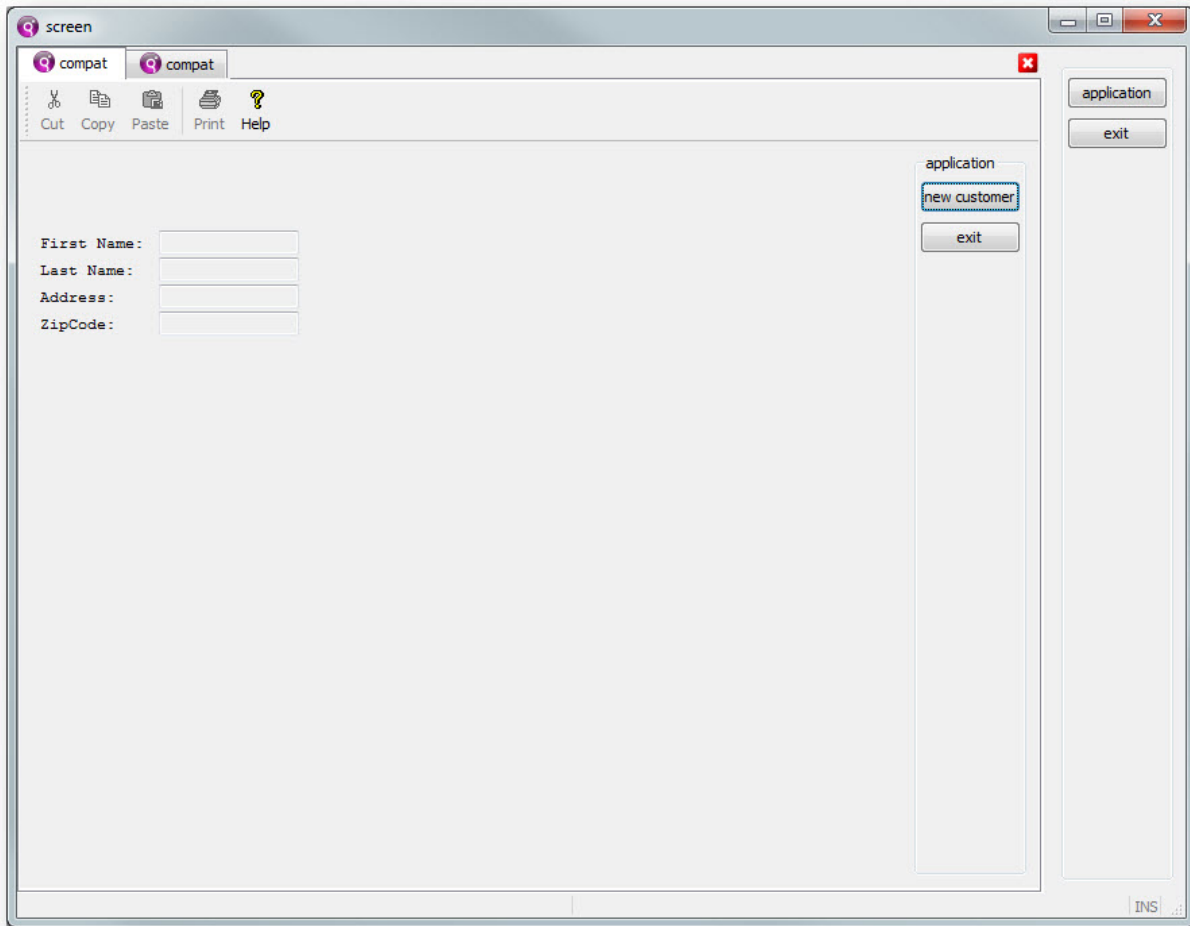


Figure 4-6. MDI container containing traditional applications

### "Pop-Tree" StartMenu support

Traditional applications can now have a StartMenu to start sub applications.

## Shortcut mechanism: Genero Desktop Client 2.40 New Features

This section describes the shortcut mechanism-related new features for Genero Desktop Client 2.40

### "Start a new shell" is no more mandatory for auto port forwarding

Introduced in 2.30, Automatic port forwarding implied to use "start a new shell" configuration option.

This is no longer the case. The feature can be used with or without starting a new shell.

## Monitor: Genero Desktop Client 2.40 New Features

This section describes the monitor-related new features in Genero Desktop Client 2.40

## Debug console configuration

Debug console output can be configured to be more or less verbose

Items displayed are now categorized and you can decide which category is displayed in the console.

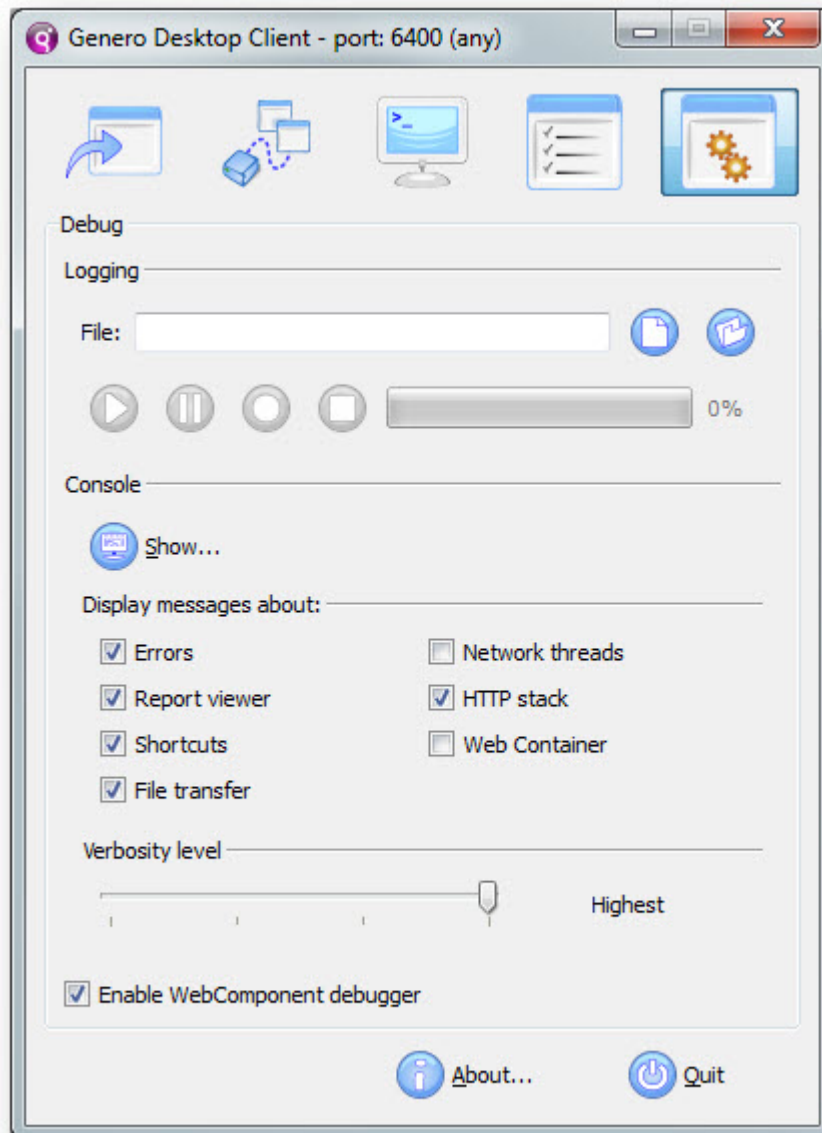


Figure 4-7. Categorize the messages to display

## Tcp server configuration

The new `--listen` command line option and Active X `setListeningMode()` API function have been added to configure the tcp server.

See “Genero Desktop Client 2.40 migration guide” on page 4-45 for more details.



## Miscellaneous: Genero Desktop Client 2.40 New Features

This section describes the miscellaneous new features for Genero Desktop Client 2.40.

### Report Viewer: improved performances

GDC is also responsible for the communication between Genero Report Engine and Genero Report writer. Performance has been highly improved and very large reports are now displayed much faster.

## Genero Desktop Client 2.40 New Features

This section describes the new features for Genero Desktop Client 2.40

These topics organize the new features of the 2.40 release of Genero Desktop Client by categories.

---

## Genero Desktop Client 2.32 New Features

These topics organize the new features of the 2.32 release of Genero Desktop Client by categories.

- Genero Features
  - Letting user manage his own image size on all screen elements
  - Show GDC version in windows for file association
- Widgets
  - Display text of toolbar items next to the icon
  - Dateedit: Better handling of misformatted date
- Misc
  - RingMenu/actionPanel: showing the beginning of the text when larger than buttons

---

## Genero Desktop Client 2.30 New Features

These topics organize the new features of the 2.20 release of Genero Desktop Client by categories.

## Genero Features: Genero Desktop Client 2.30 New Features

### Drag & Drop support

Support for Drag & Drop introduced with Genero BDL 2.30.

### WebComponent support

Support for WebComponent introduced with Genero BDL 2.30. GDC can embed a Component based on HTML / JavaScript / Flash and, via a small interface API, create a bridge between the component and 4GL. This allows you to add any html-based component to your 4GL application, such as a simple Image map which triggers a 4GL action when clicking on an area.

The GDC internal browser uses WebKit technology. If you want to use a plug-in for your WebComponent, you need to make sure that the corresponding plug-in is installed on the workstation where GDC runs.

**Note:**

- The plug-in must be compatible with WebKit ; usually Netscape plug-ins (the technology used by Mozilla Firefox) are supported by WebKit.
- If you want to use Flash inside GDC, you need to install either the Stand-Alone Flash player or the FireFox Plug-in. Having only an Internet Explorer (IE) plug-in is not sufficient, as IE plug-ins are based on a different underlying technology.
- Plug-ins are similar to external libraries that are loaded at runtime. This implies that the plug-in must be binary compatible with GDC: if you run a 64-bit GDC, you need a 64-bit plug-in. This may be an issue if you run a Flash-based plug-in on Windows, as today Adobe only provides a preview version of their Flash player on 64-bit Windows (code name "Square").

## Widgets: Genero Desktop Client 2.30 New Features

### Frozen columns for tables

You can define whether table columns should be frozen in tables. Frozen columns will always be visible. When the table is smaller than the content, a scrollbar appears.

The scrollbar will only scroll those columns that are not frozen. You can freeze columns on the left or on the right side.

```
<Style name="Table.detail">
  <StyleAttribute name="tableType" value="frozenTable" />
</Style>
```

Right click on the header of the column that should be frozen:

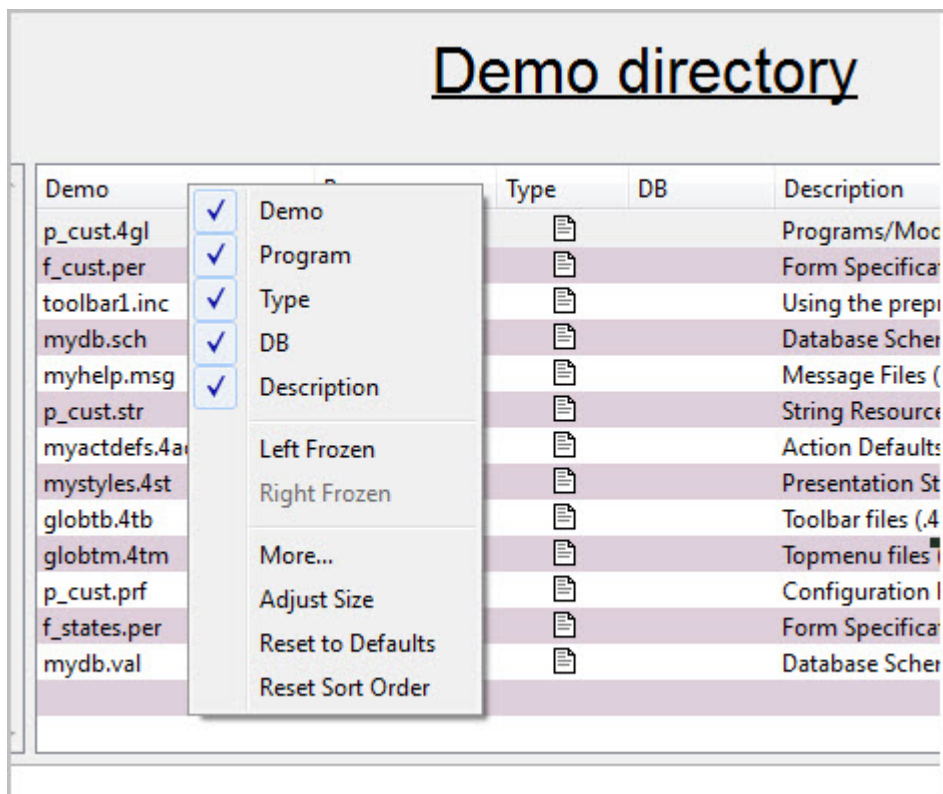


Figure 4-8. Select to freeze columns on the left or right side of the table

Now this column will always be visible; the scrolling will be done for the other columns.

This style also allows you to define columns that will be frozen on initial display:

```
<Style name="Table.detail">
  <StyleAttribute name="tableType" value="frozenTable" />
  <StyleAttribute name="leftFrozenColumns" value="1" />
  <StyleAttribute name="rightFrozenColumns" value="2" />
</Style>
```

### Text alignment in headers of tables

You can now align the headers of columns in a Table. Possible values are "left", "right", "center" and "auto". "auto" follows the justification of the content of the field which is specified by the JUSTIFY attribute in the form definition. If JUSTIFY is not specified, it follows the default data justification (for instance: left for a string field value, right for a numeric field, and so on.)

```
<Style name="Table.t1">
  <StyleAttribute name="headerAlignment" value="center" />
</Style>
```

### DateEdit: include support

DateEdit's calendar now respects the INCLUDE attribute.

Dates which are not specified in the INCLUDE attribute are greyed out and cannot be selected with the calendar.

**Example:** If include is defined as: INCLUDE=("01/01/2010" TO "12/31/2010"), the calendar displays the "normal" dates (part of the include). The dates that are not part of the include are greyed out.

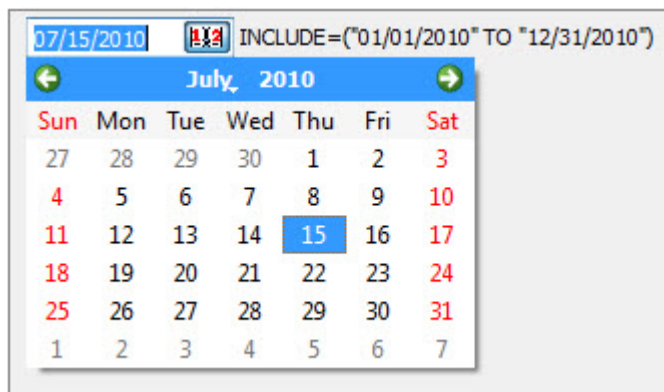


Figure 4-9. DateEdit's calendar with days disabled

#### Note:

If the field is NULL, the selected date will be the first date of the INCLUDE attribute if the current date is not included.

### Textedit / Spell Checker: Add to dictionary

End users can now add their own words to the dictionary when using spell checking.

#### Note:

- Words are not added to the dictionary file itself; a custom words file will be created in %GDCDIR%/etc (named custom\_XXX.dic where XXX is the original dic file). The current user must have write access to %GDCDIR%/etc directory. You can remove (or edit) this file to remove (or change) the custom word list.
- Dictionary loading has been reviewed: in 2.2x, each TextEdit loaded its own spell checker. Spell checkers are now shared between TextEdits during the GDC life time, which means that once a spell checker is loaded, it can be immediately used by another textedit without the delay due to loading. A dictionary is identified by the pair dic file / affix file.

### Adapt window to new form size

GDC tries to maintain the current window size. The window grows only if new content can't fit in the current size, and the GDC never shrinks a window. This is generally the best behavior, as you don't expect window size to jitter each time you change the content. Nevertheless, in the 4GL context of OPEN FORM ... DISPLAY FORM, it may make sense to shrink the window to the size demanded by the newly opened form, particularly if the new form is really small and you don't want to have a large window with lots of empty space. You can use the new "resetFormSize" form style attribute to indicate that the window must adapt its size to the newly opened form.

```
<Style name="Form.f1">
  <StyleAttribute name="resetFormSize" value="1" />
</Style>
```

### Window: Message nodes support styles

Support for message styling has been added in 2.30. When using the MESSAGE or ERROR statement, you can customize how the information is displayed. The AUI Tree defines ERROR and MESSAGE as the same node *Message*, so GDC 2.30 introduced new pseudo-selectors Message:error and Message:message. Now it is possible to:

- Apply a style to all Messages and/or Errors
- Apply a style to a specific Message or Error using ERROR "error" ATTRIBUTES(STYLE="myStyle")

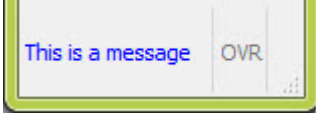
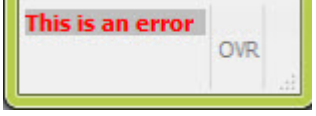
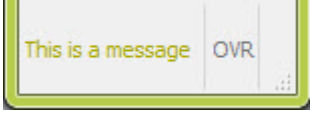
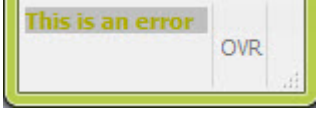
Supported Style Attributes are:

- Font style attributes (textColor, fontWeight, and so on)
- position, which is used to override the Window style property for the current message (statusbar, popup, statustip, both)
- textFormat, which is used to define whether a value should be interpreted as plain text or as html.

**Example:** 4st file:

```
<Style name="Message:error">
  <StyleAttribute name="textColor" value="red" />
  <StyleAttribute name="fontWeight" value="bold" />
</Style>
<Style name="Message:message">
  <StyleAttribute name="textColor" value="blue" />
</Style>
<Style name="Message.yellow">
  <StyleAttribute name="textColor" value="yellow" />
</Style>
```

Table 4-1. MESSAGE and ERROR examples

Statement	Result
MESSAGE "This is a message"	
ERROR "This is an error"	
MESSAGE "This is a message" ATTRIBUTES( STYLE="yellow" )	
ERROR "This is an error" ATTRIBUTES( STYLE="yellow" )	

**Note:** Similar to simple fields, tty attributes have a higher priority than styles. By default, ERROR has the tty attribute REVERSE, which explains why ERROR messages have a REVERSE background even when using styles.

### IgnoreMinimizeSetting style attribute

GDC 2.30 introduces new style attribute for Window: ignoreMinimizeSetting.

Setting this attribute to yes will make GDC ignore the minimized state stored in the settings when loading a window. This prevents a minimized window from being reopen minimized.

### SpinEdit: Support of leading 0 format

SpinEdit now accepts the FORMAT attribute (only with characters &,- and +). This allows the support of the leading 0 format. For instance, you can display 01 instead of 1:

```
SPINEDIT r1 = FORMONLY.r1, STEP=2, FORMAT="-&&";
```

### ComboBox: shortcut key for the NULL item

Delete key and Backspace key now select the NULL item in a ComboBox.

## Monitor: Genero Desktop Client 2.30 New Features

### Fgltty based on qPutty

GDC uses fgltty, a modified version of PuTTY (PuTTY). Before 2.30, fgltty was only a modified version of PuTTY that removed anything which was not used by GDC. The main drawback was that, as PuTTY is mainly developed on Windows, the support on OSX and Linux could be improved, particularly in terms of requirements (for instance, GTK2 was required for Linux, Rosetta for OSX).

Since 2.30, fgltty is based on qPutty, a Qt port of PuTTY (qputty). In addition to an easier deployment mechanism (fgltty and GDC use the same Qt libs), GDC has now access to all PuTTY options, which can be configured for each shortcut:

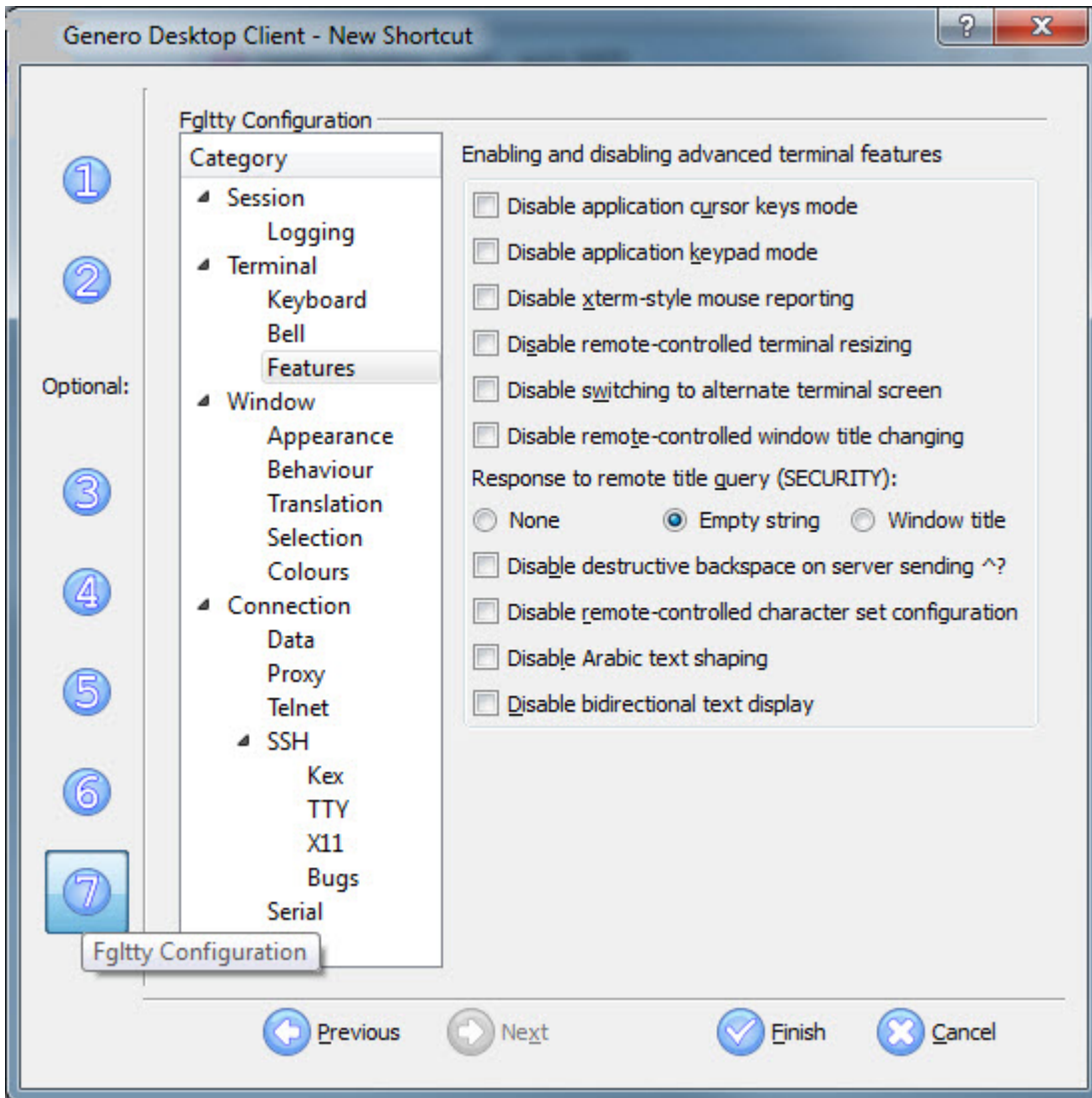


Figure 4-10. fgltty configuration window

Most of the PuTTY options have been included inside the GDC shortcut configuration. (Options which were already included are still found in the same place). For detailed information regarding PuTTY options, please have a look at PuTTY's documentation.

Please note that GDC will simply create a configuration file which will be passed to fgltty, and that fgltty relies completely on PuTTY.

### Automatic Port Forwarding

Fgltty is now able to detect a free port which can be used by the Port Forwarding mechanism.

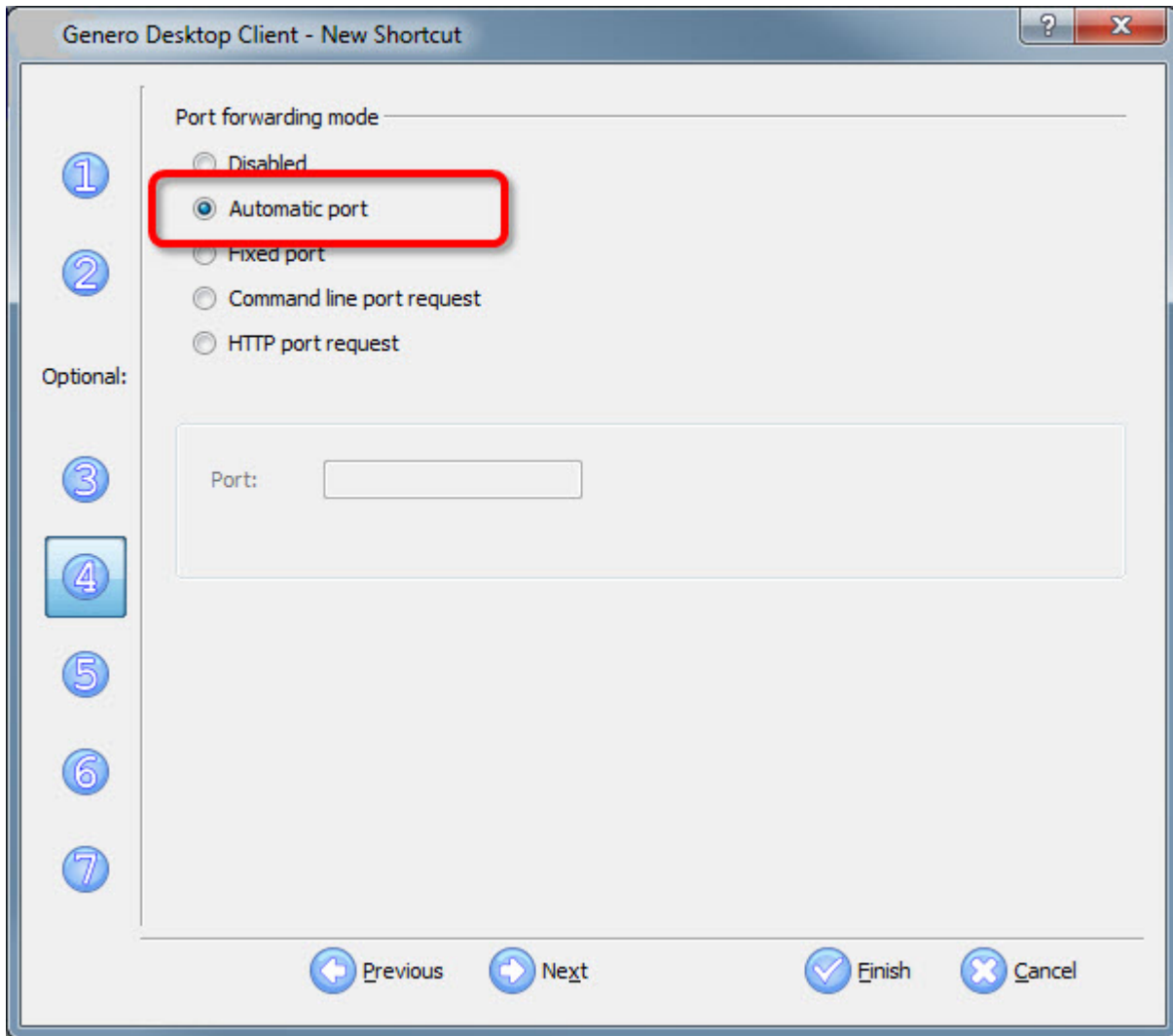


Figure 4-11. Automatic port selected under Port forwarding mode

With classic port forwarding, fglty connects once, retrieves the port to use, returns it to GDC, and GDC restarts fglty with the tunnel configuration. With automatic port forwarding, fglty establishes the tunnel during the initial connection step, finding a free port automatically. A new default Terminal String has been added: `<FGLAUTOPORT>=`, which corresponds to the message which is automatically written by fglty when selecting the auto port option.



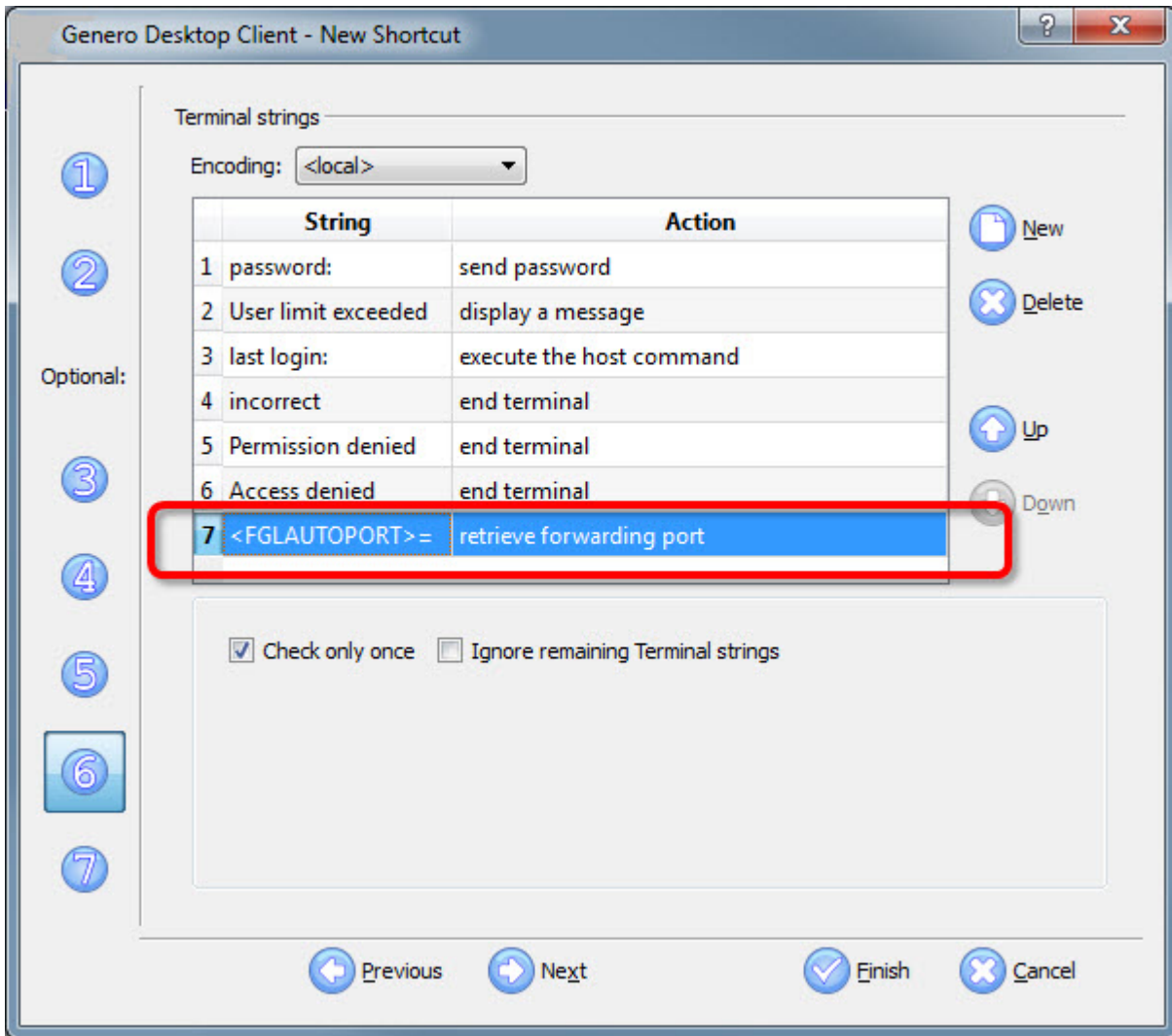


Figure 4-12. FGLAUTOPORT example

Other mechanisms are still supported if you've implemented your own port assignment system.

**Note:**

Because there is only one connection - and therefore only one start of fgltty - remote command is not passed to fgltty (@tags would be wrong as GDC has no way to know the port fgltty will use before fgltty has been started). This is why using automatic port forwarding always starts a new shell (the option will be mandatory), which means you may have to check the execute the host command connection string to match your server.

**GDC information for better support**

The GDC About box now has a **Copy To Clipboard** button, which will fill the clipboard with information that is useful when you contact your support center:

The Clipboard will contain:

- GDC version information.
- Command line used to start GDC.



- Operating system information.
- Copies of config.xml and hosts.xml.

When contacting your support center, you can copy/paste this information in your email; this should ease and speed support.

## Miscellaneous: Genero Desktop Client 2.30 New Features

### Settings for REPORT TO PRINTER (in case of DBPRINT=FGLSERVER)

When using DBPRINT=FGLSERVER and REPORT TO PRINTER, text reports are printed via GDC. Printer and fonts can be configured using the Option panel.

GDC 2.30 allows you to override these options for the current application. Two new "standard" frontcalls are available:

```
CALL ui.interface.frontCall("standard","setreportfont",
["Helvetica, Bold, Italic, 13"],[status])
CALL ui.interface.frontCall("standard","setreportprinter",
["moliere, Portrait, A4, 96 dpi, 1 copy, Ascendent, Color,
Auto"],[status])
```

#### Note:

- The values for Font and Printer are the same as the ones used in the Option panel. Simple values work (ex: "Helvetica, 18" / "Moliere"), however if you want to customize the font or the printer completely, the easiest way is to configure it with the GDC option panel and copy/paste the result. You can alternatively use "<ASK\_ONCE>", "<ASK\_ALWAYS>", "<USER\_DEFINED>" or "<USE\_DEFAULT>" as "printer" or "font" string to enforce the corresponding action.
- DBPRINT=FGLSERVER is limited and no further improvements are planned. We recommend using Genero Report Writer for your reporting needs.

### Animated Gifs support

GDC now supports animated gifs. If you load an image which is an animated gif, it will be displayed as animated.

**Important:** Animated GIFs are only available for the Image widget. They will not work as image attribute for an action panel button, a toolbar button, and so on. For performance reasons, animated GIFs are not supported in TABLE containers.

### WinMAIL: Specific Port for smtp

When using WinMail and smtp, you can now specify the smtp server port in the SetSmtp function. To define the port, use the host:port notation.

```
CALL ui.interface.frontCall("WinMail","SetSmtp", [id,
"smtp.mycompany.com:1234"],[result])
```

The default port remains 25.

### WinDDE: handling of ASCII/Wide char data

WinDDE can now dialog both with applications that require data in ASCII and with applications that require wide char data such as UTF-16. Since GDC 2.22.x, only wide char data were supported. Prior to 2.22.x, WinDDE was only able to handle ASCII. By default, WinDDE will automatically guess what is the best encoding when connecting to the DDE server. However, in some instances the returned information can be misleading. In these cases, you will have to set an optional 'encoding' parameter manually in the following DDE functions: DDEConnect, DDEExecute, DDEPeek and DDEPoke. Possible values are: "UNICODE" and "ASCII". For instance:

```
CALL ui.Interface.frontCall("WINDDE","DDEPoke",
    [prog,"Sheet1","R1C1","value","UNICODE"], [res] );
```

For more detail, see the WinDDE documentation.

---

## Genero Desktop Client 2.22 New Features

These topics organize the new features of the 2.22 release of Genero Desktop Client by categories.

### Widgets: Genero Desktop Client 2.22 New Features

#### Rich Text Editing

In previous versions, TextEdit are able to display rich text. In input, it was possible to edit rich text, but this was not straightforward. Moreover, cursor and cursor2 attributes correspond to plain text, not to the real value.

GDC 2.22 introduces rich text editing feature:

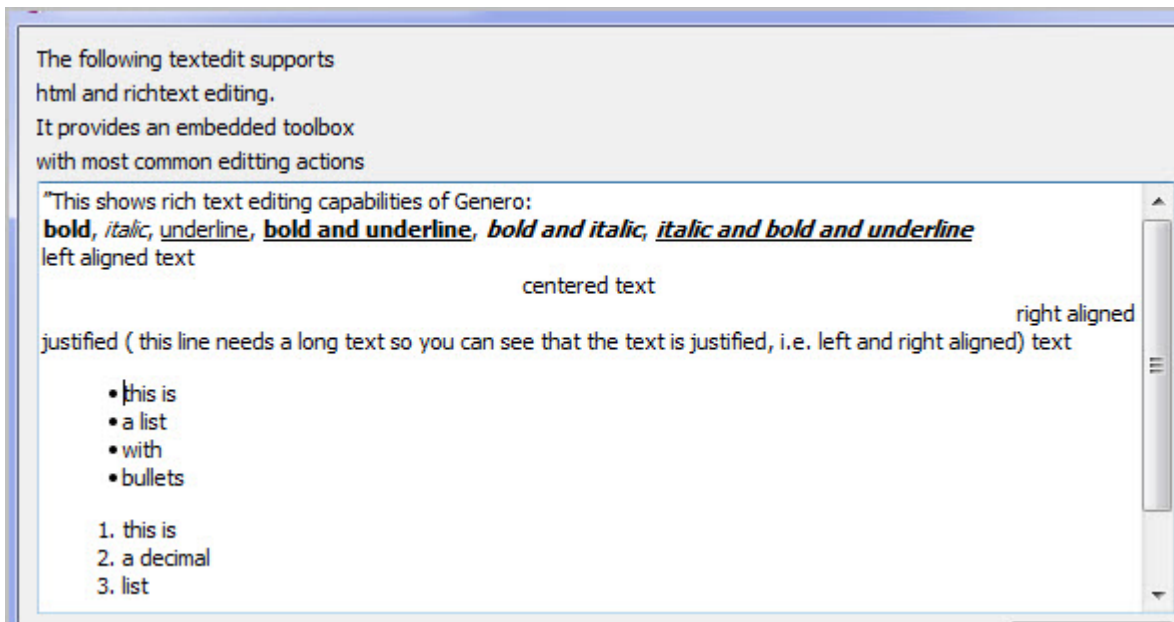


Figure 4-13. Rich text editing interface

GDC 2.22 provides:

- text format: bold, italic, underline
- paragraph alignment: left, center, right, justify
- lists: bullet, decimal
- paragraph indentation
- font size

To enable richtext editing, you need to set textFormat styleAttribute to "html" .  
<StyleAttribute name="textFormat" value="html" />

To modify your document, you can use:

- Integrated richtext toolbox

- rich text local actions

### Integrated richtext toolbox

By default, when the mouse reaches the top border of the TextEdit, a toolbox will appear. The toolbox will disappear when the mouse leaves the top border area.

This default behavior allows to keep the same height for your textedit as before - this is specially useful if you only use textedit to display rich text: the toolbox is only visible in input. If you want always display the toolbox, you can set the following styleAttribute:

```
<StyleAttribute name="textFormat" value="html" />
<StyleAttribute name="showEditToolBox" value="yes" />
```

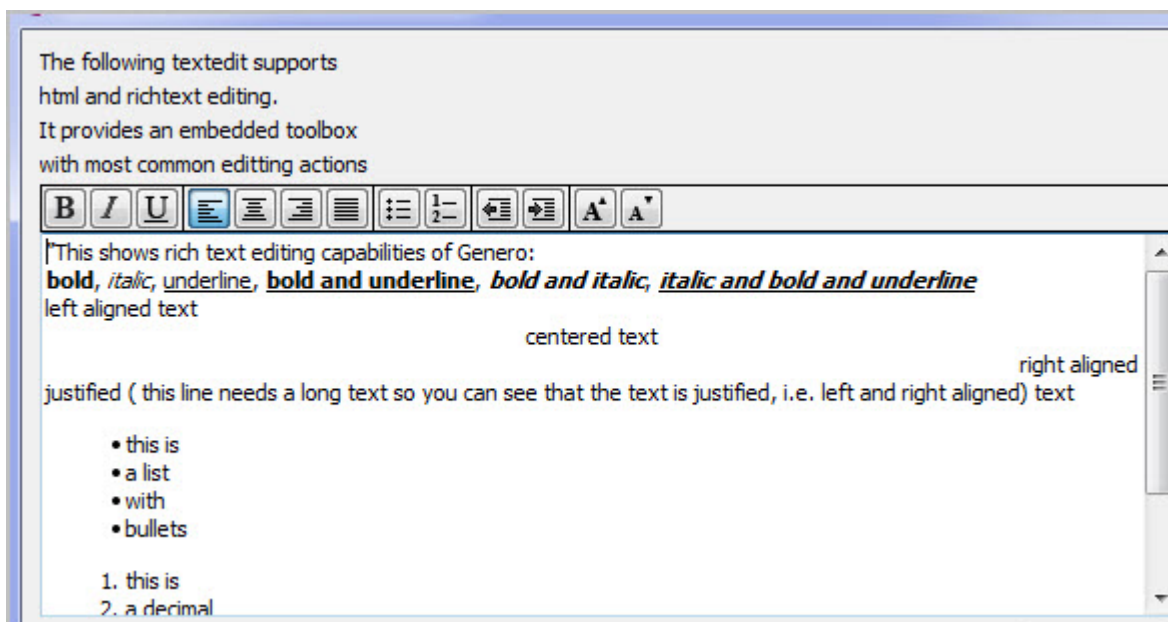


Figure 4-14. Rich text editing interface with toolbox always displayed.











#### Note:

- The textedit will be wide enough to display the toolbox entirely, even if you define a small width in the .per. Please take this in account when designing your form.
- The textedit will be high enough to display the number of lines defined in the .per (with the textedit font). This means that a textedit with a height of 1 will display toolbox and one line, which is much higher than without the toolbox.

### Rich text local actions

Besides integrated toolbox, new local actions have been created for each rich text capability. As any local action, you can configure accelerator keys, and bind them to action views like ToolBar buttons.

Table 4-2. Local action names, accelerators, and icons

Name	Default Accelerator	Icon Name	Icon
richtextbold	control-b	textbold	<b>B</b>
richtextitalic	control-i	textitalic	<i>I</i>
richunderline	control-u	textunder	<u>U</u>
richtextalignleft	control-l	textleft	
richtextaligncenter	control-e	textcenter	
richtextalignright	control-r	textright	
richtextalignjustify	control-j	textjustify	
richtextlistbullet	None	textlistbullet	
richtextlistdecimal	None	textlistnumbered	
richtextdecreaseindent	None	textindentdecrease	
richtextincreaseindent	None	textindentincrease	
richtextdecreasefontsize	None	textfontsizeup	
richtextincreasefontsize	None	textfontsizeup	

Then you can hide the toolbox using the following styleAttribute:

```
<StyleAttribute name="textFormat" value="html" />
<StyleAttribute name="showEditToolBox" value="no" />
```

**Important:**

- We are not generating html code by ourselves. We are using a component dealing with rich text which provides a "toHTML" export. As we have nearly no way to influence the export, and are completely dependent on the component, future versions of GDC may behave differently if the component provider decides to change the export. Should this occur, we will add some entries in the Migration Guide.
- cursor and cursor2 attributes are now following better html code, but they are still not 100% corresponding. For instance, if you load an html file with a hidden part, cursors will be wrongly set. We recommend using cursor and cursor2 with care when textFormat is set to html.

---

## Genero Desktop Client 2.21 New Features

These topics organize the new features of the 2.21 release of Genero Desktop Client by categories.

### General category: Genero Desktop Client 2.21 New Features

#### GDC 2.21 is now compatible with FGL 2.11 Runtime

You can now run your 2.11 application with GDC 2.21, your application behavior will be the same as GDC 2.11. This can be useful if you need to mix FGL 2.11 and 2.2x installations: you only need to install GDC 2.21.

**Note:** As GDC 2.21 has only one rendering engine, all applications will have the look and feel introduced in 2.20. However, the application logic will be the same and it will work the same way as GDC 2.11.

### Internal HTTP stack has been rewritten

This enables pre-2.20 lost features like Kerberos Support, NTLM single Sign-on or Client certificate.

GDC is also able to retry when network errors occur. This was already the case in previous (late MR) versions, but now it may retry in more situations (ex: in case of HTTP error 500). This can be useful if your network is not too reliable and sometimes messages may be discarded before reaching GAS or returning to GDC. You can now configure how GDC will retry:

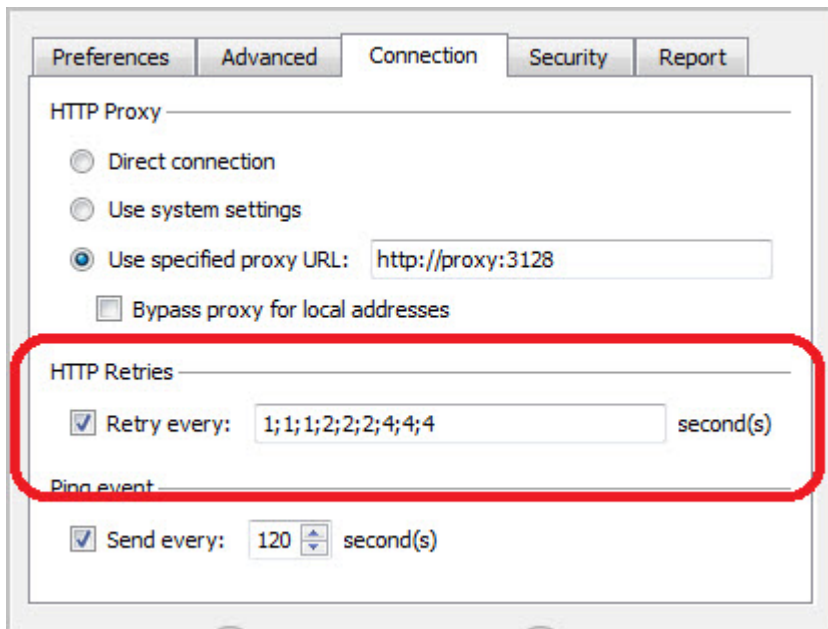


Figure 4-15. HTTP Retries section of Connection tab

The entry "1;1;1;2;2;2;4;4;4" means that GDC will retry 9 times. GDC will wait 1 second between the first three errors, then 2 seconds between the next three, and 4 seconds between the last three. GDC will then wait a maximum of 21 seconds before showing an error message.

#### **Note:**

This is the time GDC waits AFTER the system / network returns an error, not the complete waiting time. For instance, if the system needs time to answer (ex: connection timed out), GDC will wait:

1. for the system
2. for the configured wait time

GDC now shows some information in the systray icon when there is an HTTP connection issue.

## Windows: Genero Desktop Client 2.21 New Features

### Automatic scrollbars

Maximized windows show scrollbars around the form if the window is larger than the desktop available size.

This old feature has been completely reviewed to now work also when the window is not maximized. GDC will then automatically, when the window is larger, fit the size of the window to the desktop size (height or width) and display scrollbars around the form.

**Note:** This feature should be used carefully. A Desktop application is not a web application, and having scrollbars (especially horizontal ones) around the form is not common. As GDC will try to always show the current field, this may lead to lots of scrolling when you move from one field to another if the fields are not all visible.

We strongly recommend that you adapt your forms to the smallest desktop size you target; automatic scrollbars should only appear for "accident" cases.

If you prefer avoid automatic scrollbars and retrieve the behavior of previous versions (only getting scrollbars when the current window is maximized) you can use the following style attribute:

```
<Style name="Window.myWindow">  
  <StyleAttribute name="formScroll" value="no">  
</Style>
```

To achieve automatic scrollbars in a more stable way, the action frame (menu/dialog) has been reviewed. The new look is very slightly different, but the main behavior is the same.

- Navigations button are different if you are on the first or last row.
- A "plus" button has been added to display in one click all remaining items.
- A little animation shows scroll direction.

### A window can be started as minimized

As well as "maximized", windows can be started minimized using minimized as the value for the "windowState" style attribute. This can be useful if your program loads the first window, and then uses a second one for the first dialog ; you can now have the first window minimized.

## Widgets: Genero Desktop Client 2.21 New Features

### DateEdit presentation style showCurrentMonthOnly

DateEdit now has a presentation style named "showCurrentMonthOnly". This style configures whether the calendar shows only the current months, or displays (in light grey) days of the previous and next month. The default behavior is that the days of the previous and next month are visible.

Default style definition:

```
<Style name="DateEdit.onlyCurrentMonth">  
  <StyleAttribute name="showCurrentMonthOnly" value="no">  
</Style>
```

Style definition showing previous and next months (showCurrentMonthOnly="yes"):

```
<Style name="DateEdit.withPreviousAndNext">
  <StyleAttribute name="showCurrentMonthOnly" value="yes">
</Style>
```

### **SpinEdit new attributes**

Support for SpinEdit new attributes: valueMin and valueMax to set a range on the SpinEdit.

### **Image supports alignment attribute**

Image field now supports style attribute "alignment" to define where the picture should be located when the container (widget) is bigger.

Values can be combination of { top, bottom }, { left, right } and center ; for instance "bottom left":

```
<Style name="Image.centered">
  <StyleAttribute name="alignment" value="center"/>
</Style>
<Style name="Image.bottomright">
  <StyleAttribute name="alignment" value="bottom right"/>
</Style>
```

When the attribute autoscale is set, alignment will only be applied on the direction which does not fill the widget. By default, alignment will be like "top left" in previous versions.

### **Window supports toolBarDocking**

Window supports now style attribute "toolBarDocking" to define if the toolbars are movable and floatable.

Values can be "yes" (default), "no".

```
<Style name="Window.style1">
  <StyleAttribute name="toolBarDocking" value="no"/>
</Style>
```

## **Presentation Styles: Genero Desktop Client 2.21 New Features**

### **Non-integer point font size**

You can now set a point font size with a non-integer value, for instance 8.3pt:

```
<Style name=".big">
  <StyleAttribute name="fontSize" value="18.5pt">
</Style>
<Style name=".small">
  <StyleAttribute name="fontSize" value="8.3pt">
</Style>
```

This fixes a small issue: the default font size on Windows is 8.25, but as GDC only managed integer font size, the default font size was set to 8.

## **Monitor: Genero Desktop Client 2.21 New Features**

### **Connection tab displays application title**

Previous versions only displayed the application name (usually 42r name) ; now the tab shows the text (if set using ui.interface.setText()). Each application will also show in the tooltip the list of all open windows.

### **User limit exceeded feedback possible**

User Limit exceeded is now a default terminal string, so the end user has feedback if his application can't start because of a license issue:



## Debugging information added

Debugging information has been added to the debug console when a shortcut starts.

Analyzing this log may help you to understand what GDC did, and why a connection may have failed.

## Windows 7 support

GDC 2.21 is now supported on Windows 7 platform.

## Bitwise WinSShd 5 support

Bitwise modified the way their WinSShd server uses the terminal, which leads to some issues with GDC Terminal Strings.

GDC has been adapted to work with WinSShd 5.0.9 (some changes in the server have been done by Bitwise too, so you'll need to upgrade the server part to at least version 5.0.9.)

## FrontEnd functionCall: Genero Desktop Client 2.21 New Features

### hardcopy frontCall available

"Hardcopy" is now available as a frontcall. It allows you to print a screenshot of the current window.

```
CALL ui.interface.frontCall("standard","hardcopy",[],[ret])
```

There is an optional parameter to indicate if the screenshot must be adapted to the page size:

```
CALL ui.interface.frontCall("standard","hardcopy",[1],[ret])
```

In MDI container cases or MDI child cases, calling this front call will print the container (except when the current window is modal).

### launchurl frontCall signature

"launchurl" frontCall signature has the same signature in GWC and in GDC

In 2.20, GWC was able to manage a parameter which was not handled by GDC. This forced you to write code like:

```
IF ui.Interface.getFrontEndName() == "GWC" THEN
  CALL ui.interface.frontCall("standard","launchurl",[url, mode],[ret])
ELSE
  --GDC does not support "mode"
  CALL ui.interface.frontCall("standard","launchurl",[url],[ret])
END IF
```

Now GDC silently ignores the parameter so you can use the same code with both front-ends:

```
CALL ui.interface.frontCall("standard","launchurl",[url, mode],[ret])
```

## Miscellaneous: Genero Desktop Client 2.21 New Features

### Hardcopy available for MDI child windows

Windows Only: The HardCopy menu item in system menu was previously available for SDI windows only. The HardCopy menu item is now available in the system menu for MDI child windows.



## Windows now uses MSI installer system

Windows now uses the MSI installer system.

## Improved Desktop integration

- .gdc files are associated with GDC to be run directly in your favorite explorer (Windows, Linux).
- Linux installer creates entries in your desktop start menu.

## Experimental features: Genero Desktop Client 2.21 New Features

### CAUTION:

This version contains a few experimental features. Experimental features are available in the product, but:

- they are likely to be changed in future versions, or even simply removed from the product.
- they are not supported. We won't be able to fix all reported issues; most of the time, this is due to current technical limitations.
- they may not work 100%, not on all platforms, and likely not with all Front-Ends.

These experimental features are provided 'as is', so you can play with them and return your feedback, but we may not be able to fix an issue, or we may even remove the functionality if a serious side effect / performance issue is seen. They are usually based on unfinished work, or on third party tools on which we can't rely 100%. But, we believe it's nice to have them in the product and to show you today what we're likely to be able to do tomorrow. You can use them freely, but at your own risk.

### Flash support

It is now possible to see a Flash application in the pages you display with the Integrated browser.

**Note:** This feature uses "Netscape Plugin" technology, which is also used by Mozilla Firefox or Google Chrome. So you need to have Firefox, Chrome plugin or stand-alone Adobe flash player installed. Having Microsoft IE plugin only is not enough.

With this new feature you can now watch your favorite YouTube video in your 4GL application.

### Compositing

Windows Vista and Windows 7 introduced Compositing.

This allows you to make some fancy effects with window transparency. If you want a semi-transparent window in GDC, you may use the "blurBackground" style attribute:

```
<Style name="Window.semitransparent">
  <StyleAttribute name="blurBackground" value="yes"/>
</Style>
```

See the classic demo application with a semi-transparent background, running on Windows 7. This will not work on "old" windows (XP and before), but may work on Linux depending on the windows manager capabilities.

## Windows Vista introduced a Command Link Button

Windows Vista introduced a Command Link Button: a push button with a nicer design, showing the comment directly. When the mouse goes over the button, a nice gradient effect occurs.

If you want a Command Link Button in GDC, you may use "commandLink" as the value for the "buttonType" style attribute:

```
<Style name="Button.commandLink">
  <StyleAttribute name="buttonType" value="commandLink"/>
</Style>
```

You can now add a Vista (or Windows 7) Command Link Button style in your application, and use the mouseover effect.

This will work on Vista / Windows 7 only. On other systems it displays a simple button with the text and the comment, but without the nice mouseover effect.

## Save and restore current window size

Two new frontcalls have been added in the standard frontcall library:

- **storeSize** asks GDC to store the current size of the current window.
- **restoreSize** asks GDC to restore the stored size.

This allows you to create the classic GUI with Show/Hide details.

When **show** is clicked, the window grows to show more information. When **hide** is clicked, the window returns to its original size.

```
ON ACTION details
  IF state = 1 THEN
    CALL f.setElementHidden("g2",1)
    CALL f.setElementText("details",&Show details")
    CALL ui.interface.frontCall("standard","restoreSize",[200],[ret])
    LET state = 0
  ELSE
    CALL ui.interface.frontCall("standard","storeSize",[],[ret])
    CALL f.setElementHidden("g2",0)
    CALL f.setElementText("details",&Hide details")
    LET state = 1
  END IF
```

The restoreSize frontcall takes an optional parameter to define the delay (in milliseconds) used to revert the window size. The window will then smoothly shrink or grow to reach the saved size instead of having its new size immediately.

### Note:

- Calling restoreSize without calling storeSize, or on a different window, has no effect.
- The stored size is a desired size; the layout has always higher priority. For instance, if the saved size is 800x600 and the content of the window is 1024x768, GDC will not be able to shrink to the expected size.

---

## Genero Desktop Client 2.20 New Features

These topics organize the new features of the 2.20 release of Genero Desktop Client into categories.

## General Category: Genero Desktop Client 2.20 New Features

### Internal library used is now Qt4

Qt4 is now the internal library used for the Genero Desktop Client.

### SVG image format added to supported image format list

SVG is supported in two ways:

- If an SVG image is displayed to an IMAGE field (or static IMAGE), an SVG renderer is used. If the widget is resized (according to STRETCH and AUTOSCALE attributes), the image is resized without resize artifacts.
- On other items, as they can't be resized, the SVG image is used as a pixmap, as well as all other image formats.

### Image Cache

Images used in GDC are now copied and stored locally, to accelerate image lookup. The cache can be configured (enabled, cache size, and so on) in the 'Advanced' Option tab:

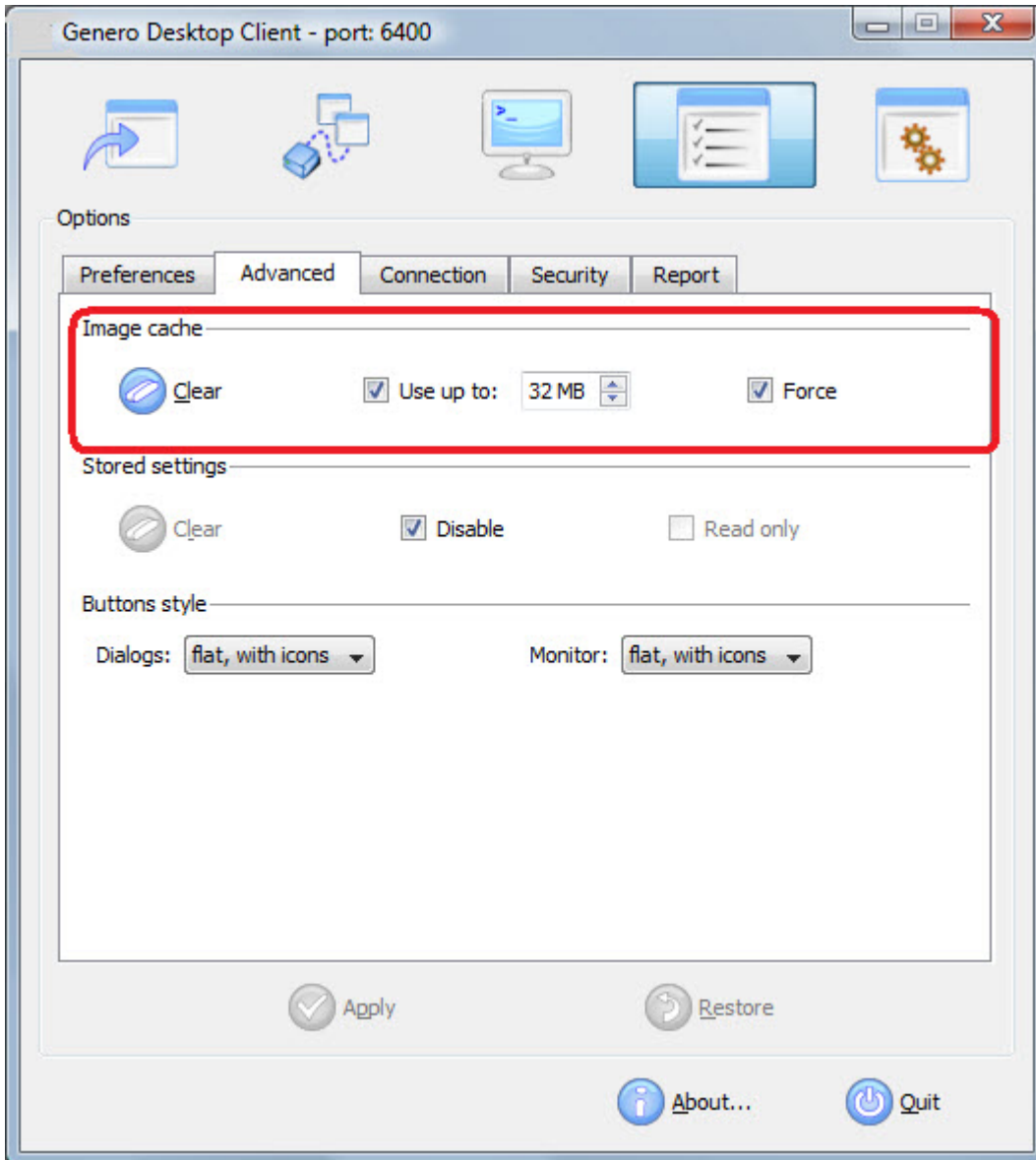


Figure 4-16. Image cache section of the Advanced tab

A new style has been introduced to manage the cache (disk cache and memory cache): imageCache. It can be applied to any item using images and defines whether GDC must cache the image (based on the url).

Values can be yes (cache is used) or no (cache is not used). The default behavior depends (as in previous versions) on the item type:

- IMAGE fields are not cached.
- All other items (button, toolbar items) are cached.

## Monitor / Shortcut mechanism: Genero Desktop Client 2.20 New Features

### A customized login box can be created for shortcuts

Find details in the Shortcut wizard section

## Command Line option for ignore settings

"-i" or "--ignoreSettings" command line argument forces ignore settings. It has the same effect as the checkbox in the Advanced Tab, but on the command line.

## Read-Only Stored settings

A read only stored settings option has been added on the Advanced Tab.

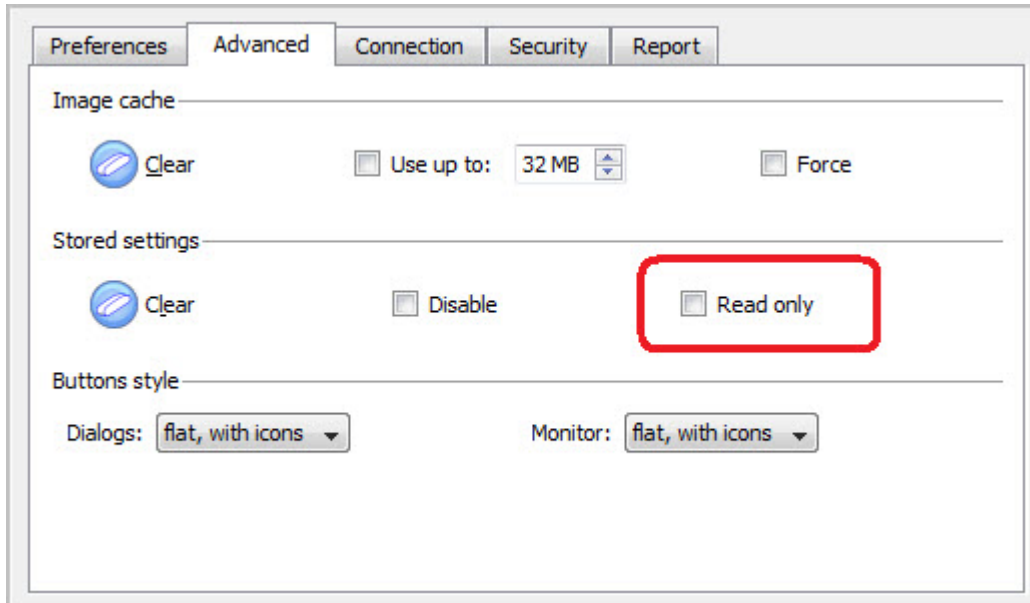


Figure 4-17. Read-only check box on the Advanced tab

Settings are applied when loading a form, but they are not modified when closing the form.

## Command Line Option for recording log

"-r [filename.log]" or "--logrec [filename.log]" to start recording a log immediately. If no filename is provided, the log will be named `<hostname>_<date>_<time>.log`.

## SSH2 is default protocol type when -T is not set

When starting a shortcut with the -s option, -T defines the protocol used to connect to the remote host. SSH2 is now used instead of SSH when -T is omitted.

## Import / Export shortcut and .gdc files

Shortcuts can be exported as and imported from a .gdc file. GDC can also be started with a .gdc file and the shortcut starts directly. You can see more details [here](#).

## Security Level in http

When connecting directly to the runtime system, some tests are done, depending on the Security level, to accept the connection.

When connecting via Application Server, the security level was ignored. Now, when using GAS 2.20, GDC 2.20 and DVM 2.20, the security level is also managed to be sure the application you have on GDC is the one you started.

## Miscellaneous

- Several Connections and Terminals can be selected and closed (instead of closing them one by one)
- Default Proxy and Kerberos Realm can be defined in the Options / Connection panel. This information is used when GDC connects to GAS as well as when GDC is looking for an image.

## FrontEnd functionCall: Genero Desktop Client 2.20 New Features

### functionCall Standard "getwindowid" added

Returns the window manager's id for a given Window.

```
LET auWinID = getCurrentWindowID()
CALL ui.interface.frontCall("standard","getwindowid",[auWinID],[ret])
IF ret <> -1 THEN
  CALL my_function(ret)
END IF
```

### New parameters for functionCall Standard "feinfo" added

- osversion returns the version of the OS (Windows / OSX only).
- numscreens returns the number of screens available.
- screenresolution (with optional screen number parameter) returns the screen resolution.
- fepath returns the path of GDC executable.

## Widgets: Genero Desktop Client 2.20 New Features

### Treeview widget support

You can now add a TreeView widget in your 4GL application, based on a simple DISPLAY ARRAY.

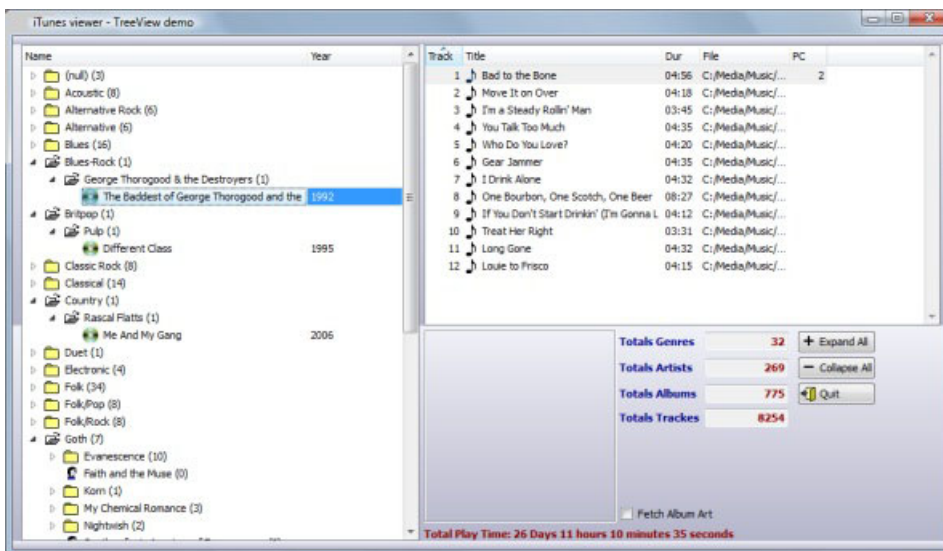


Figure 4-18. Treeview widget

## Labels and TextEdits: Hyperlinks with html textFormat

HTML hyperlinks are now managed in Labels, when styleAttribute "textFormat" is set to "html" . Links are open with the default link handler (default browser for http links, for instance).

## TextEdit: Integrated search facility

TextEdit can have an integrated search facility with style "integratedSearch" . Pressing Control+F will show the search bar.

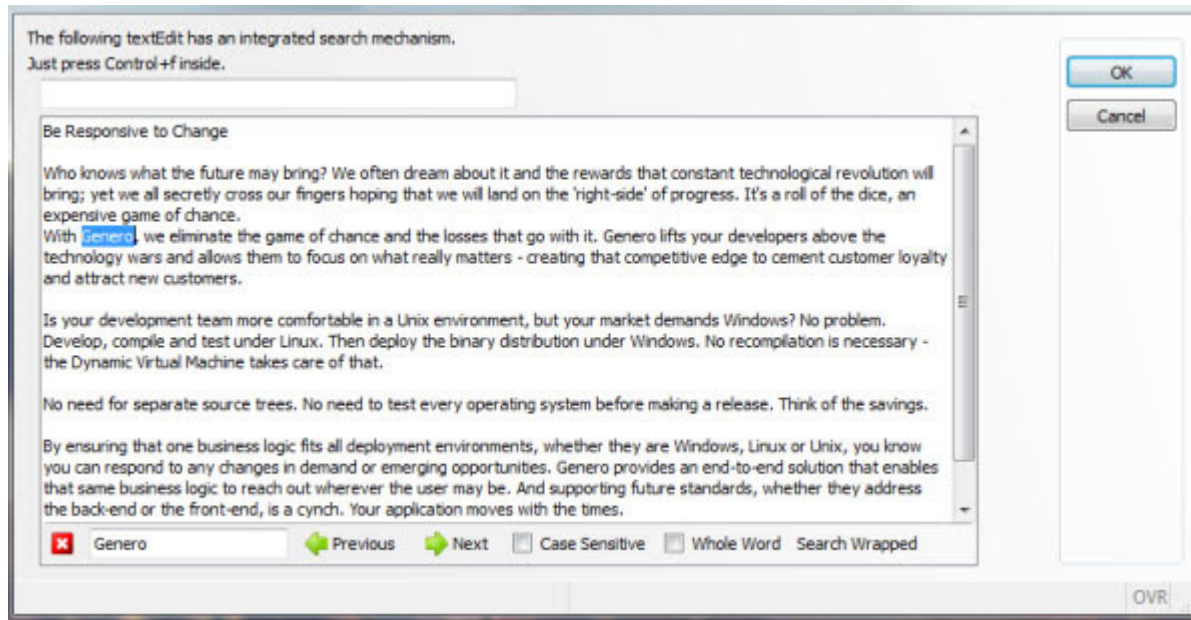


Figure 4-19. Integrated search

F3 and SHIFT+F3 search for the next and previous occurrences (as well as the previous / next buttons); when you start typing in the search area, the text is selected. A "Search Wrapped" message is displayed if you reach the end of the document. You can optionally search case-sensitively, or for only whole words.

```
<Style name="TextEdit.searchOn">  
  <StyleAttribute name="integratedSearch" value="yes" />  
</Style>
```

## TextEdit: Spell Checker

A spelling checker is included for TextEdits. It uses LGPL Hunspell (see <http://hunspell.sourceforge.net/>) which is the default spell checker of OpenOffice.org or Mozilla tools.

You need to provide GDC the two dictionary files needed for each language. These files can be downloaded from <http://wiki.services.openoffice.org/wiki/Dictionaries>.

**Important:** You need to use dictionary files available for OpenOffice.org. GDC is not supporting dictionary files for OpenOffice.org 3.x

The files can be stored on the Runtime System side and can be uploaded to GDC using a simple FGL\_PUTFILE. Then specify in the style the two files for the "spellCheck" StyleAttribute, using the following format:

```
"<affix file>|<dictionary file>"
```



By default, files are read from where you're executing GDC. You can, however, specify an absolute path:

```
<Style name="TextEdit.spellUs">
<StyleAttribute name="spellCheck"
  value="file:///c:/dics/en_US.aff|file:///c:/dics/en_US.dic" />
</Style>
```

GDC will underline unknown words, and a right click on the unknown word will add the list of suggestions for this word to the contextual menu.

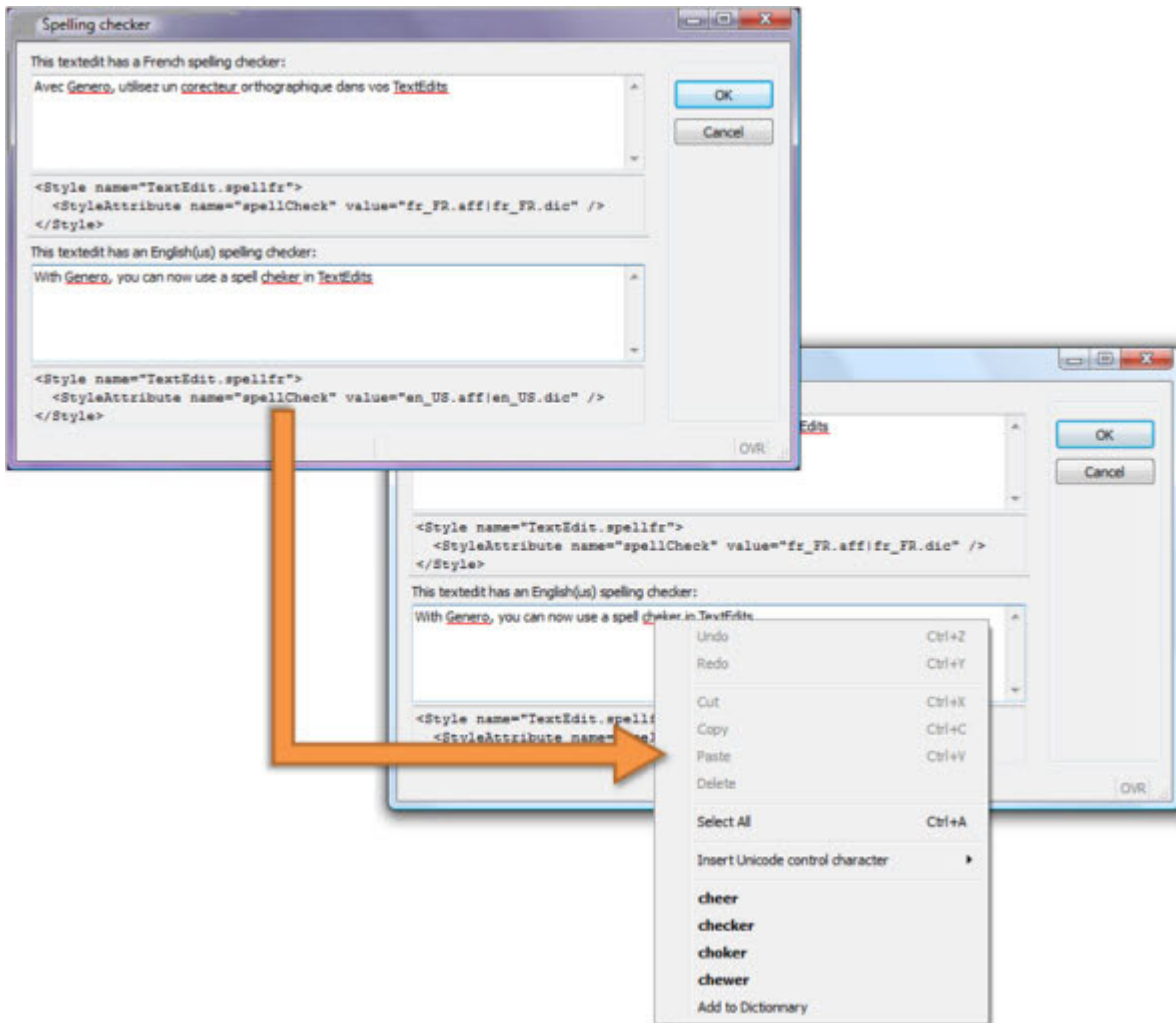


Figure 4-20. Spell checker window flow

```
<Style name="TextEdit.spellUs">
<StyleAttribute name="spellCheck" value="en_US.aff|en_US.dic" />
</Style>
```

**Note:**

- Loading the dictionary takes time (Approx. 5 seconds for the French dictionary for instance). This is why the spell checking may not be immediate when loading the form.
- If there is an error loading dictionary files, the checking will simply not be done. No error will be reported on the 4GL side.



## Folders: StyleAttribute for position

Folder nodes support a "position" style attribute. Values can be: "top" (default), "left", "right", "bottom" and defines the position of the tabs.

## Action Frame (Action Panel and Ring Menu): StyleAttribute for decoration

New Window style attributes have been introduced to define the decoration for the ActionFrame: "actionPanelDecoration" and "ringMenuDecoration". The attributes define whether the Groupbox decoration is visible. These attribute can also define whether the action frame can be "undocked" by the user and moved around the window. Values can be:

- "auto" (default): left and right action frames have groupbox decoration, but top and bottom haven't.
- "yes": the groupbox decoration is always drawn.
- "no": the groupbox decoration is never drawn.
- "dockable": the action frame can be undocked, and has therefore a dedicated decoration.

### Example:

```
<Style name="Window.dockAF">
  <StyleAttribute name="actionPanelDecoration" value="dockable" />
  <StyleAttribute name="ringMenuDecoration" value="dockable" />
</Style>
```

## Menu: Style Attribute for popup menu position

Menu "popup" supports a "position" style attribute. Values can be:

- "cursor" (default): at the mouse cursor position
- "field" : on the current field
- "center" : in the center of the screen
- "center2" : in the center of the current window

## Menu: Menu and mouse wheel

You can now use the mouse wheel to change the focus of the menu.

## Button: ButtonType style attribute to customize Buttons

The Button node supports a "buttonType" style attribute. Values can be:

- normal (default)
- link : the button is drawn as an HTML hyperlink. In contrast to the Label Hyperlink support, clicking on a "link" button does not start the default browser, but triggers the corresponding action, like a normal Button.

### Example:

```
<Style name="Button.link">
  <StyleAttribute name="buttonType" value="link" />
</Style>
```

## Tables: Genero Desktop Client 2.20 New Features

**Tables: MultiSelection in DISPLAY ARRAY:** Different lines can be selected using the usual key combination (shift, control, and so on).

```
DISPLAY ARRAY ar TO sr.*
  BEFORE DISPLAY
  CALL DIALOG.setSelectionMode( "sr", 1 )
END DISPLAY
```

**Tables: Sort in INPUT ARRAY:** Columns can be sorted in INPUT ARRAY.

**Tables: Picture Flow design:** The TABLE widget can be used to render a Picture Flow, a widget to display images with an animated transition effect. The first column of your array will be used for the image file name:

```
<Style name="Table.PictureFlow">  
  <StyleAttribute name="tableType" value="pictureFlow" />  
</Style>
```

**Tables: resizeFillsEmptySpace styleAttribute:** When you resize a table and make it wider than the size of the content, an empty space can appear on the right side. Set the styleAttribute `resizeFillsEmptySpace` to "yes" and the last column will be automatically resized to fill the empty space:

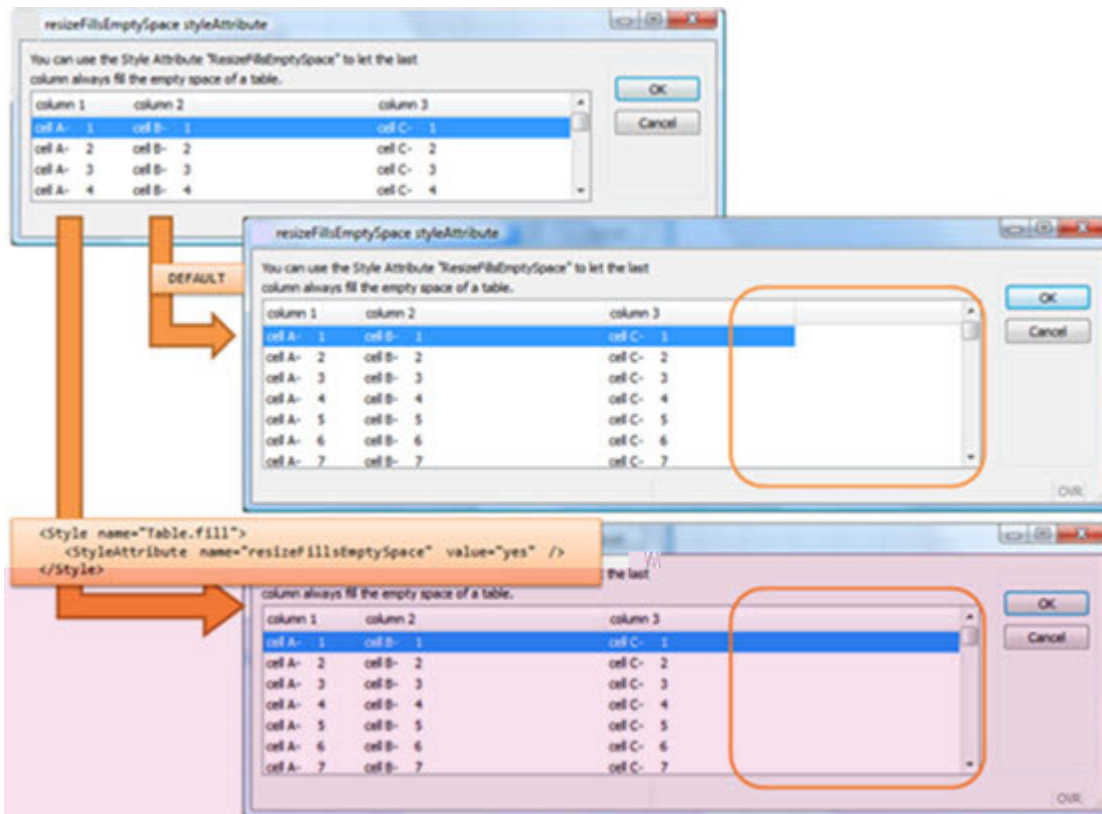


Figure 4-21. `resizeFillsEmptySpace`: Before and After

## Windows: Tabbed MDI

MDI Containers can now be displayed as "Tabbed MDI" (like FireFox / IE7):

Two StyleAttributes allows you to configure this: `tabbedContainer` defines whether the Container should be "tabbed". `tabbedContainerCloseMethod` defines how to close the current page. Values can be:

- "container" (default), the container has a close button on the top right corner, which closes the current tab.
- "page", each page has its own close button.
- "both", each page and the container have a close button.
- "none", no close button is shown.

The close button is enabled depending on the window style attribute.

## Action mechanisms: Genero Desktop Client 2.20 New Features

### Local Action priority

To adapt to Multiple Dialogs and ON ACTION ... IN FIELD, the LocalAction mechanism has been slightly revised. In prior versions, an Runtime System Action always had higher priority than LocalActions. This means that if you have an inactive Runtime System Action having the same name as a LocalAction, all ActionViews (Buttons, toolbar items...) are disabled. In 2.20, when the Runtime System Action is disabled, the state of the ActionViews will be the LocalAction state.

### Qualified Local Actions

GDC now creates *qualified local actions*, i.e. it creates local actions prefixed by the screen record, for instances `r1.nextrow`. This allows you to create ActionViews bound to a specific screen record, such as a dedicated navigation panel for a specific table.

## Experimental features: Genero Desktop Client 2.20 New Features

### CAUTION:

This version contains a few experimental features. Experimental features are available in the product, but:

- they are likely to be changed in future versions, or even simply removed from the product.
- they are not supported. We won't be able to fix all reported issues; most of the time, this is due to current technical limitations.
- they may not work 100%, not on all platforms, and likely not with all Front-Ends.

These experimental features are provided 'as is', so you can play with them and return your feedback, but we may not be able to fix an issue, or we may even remove the functionality if a serious side effect / performance issue is seen. They are usually based on unfinished work, or on third party tools on which we can't rely 100%.

But, we believe it's nice to have them in the product and to show you today what we're likely to be able to do tomorrow. You can use them freely, but at your own risk.

### Look and Feel

The "lookAndFeel" style attribute allows you to customize your application.

All widgets used by GDC can follow a given style, which defines how a widget must be drawn on the screen. By default, it uses the system platform style. The internal style used by GDC will be modified.

If you're a good C++ developer, you can build your own style, following Qt's guidelines.

A few built-in styles are provided; the list, which depends on the platform, is available in the about box. **Example**

In your 4st file:

```
<Style name="Window.Office">
  <StyleAttribute name="lookAndFeel" value="dotnetoffice" />
</Style>
```

## Form Layout

The Form Layout is a style applied to a GRID that lets GDC display the content of the section, ignoring the .per alignment and so on. Simple fields are bound to their labels using the TITLE attribute.

### Example

The GRID section only contains the fields:

```
GRID (style="flayout")
{
  [firstname  ]
  [lastname   ]
  [address    ]
  [           ]
  [postcode   ]
  [city       ]
}
END
```

The ATTRIBUTES section defines the TITLES:

```
ATTRIBUTES
EDIT firstname = FORMONLY.firstname, TITLE("&First Name:");
EDIT lastname  = FORMONLY.lastname,  TITLE("&Last Name:");
TEXTEDIT address = FORMONLY.address,  TITLE("&Address:", stretch=y;
EDIT postcode   = FORMONLY.postcode,  TITLE("&ZipCode:", PICTURE="#####");
COMBOBOX city   = FORMONLY.city,      TITLE("&City:",
  ITEMS=("Paris","Strasbourg","Erfurt");
```

The 4st style file specifies the form layout:

```
<Style name="Grid.flayout">
  <StyleAttribute name="layoutType" value="form" />
</Style>
```

A form will be created, associating fields and their title. The alignment of the titles depends on the platform; if the title contains "&", pressing ALT + the following letter will set the focus on the attached field.

## Integrated Browser

TextEdits can display more or less complex html data, but can't act as a real browser.

With styleAttribute imageContainerType Image style attribute set to "browser", your image container becomes a browser. Instead of setting an image name, you can set a URL.

### Example

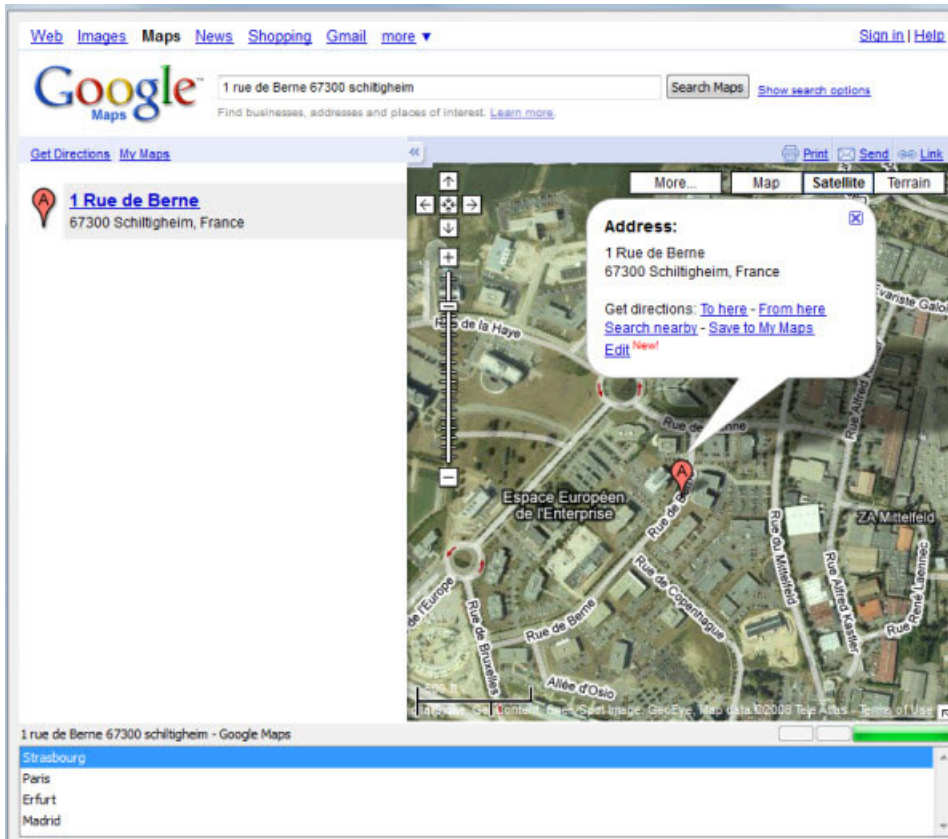


Figure 4-22. Google maps screenshot

Here is the per file:

```
LAYOUT (TEXT="Google Maps")
  GRID
  {
    [map ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
    [
    ]
  }
  <T
  t >
  [location ]
  [location ]
  [location ]
  }
  END
  END
  ATTRIBUTES
  IMAGE map = FORMONLY.map, STRETCH=BOTH, style="map";
  TABLE t : WANTFIXEDPAGESIZE, style="list";
  EDIT location = FORMONLY.location;
  END
  INSTRUCTIONS
  SCREEN RECORD sr(location);
```

Here is the corresponding .fst file:

```
<Style name="Image.map">
  <StyleAttribute name="imageContainerType" value="browser" />
</Style>
```

**Important:** This feature uses the current WebKit Open Source project; we use the version provided with Qt and we've no control over it. The aim is strictly to be able to display some HTML/Rich Text, not the most complicated pages of the web. Indeed, this feature comes with limitations, such as no Java™ support and no activeX support. We expect the version to be better supported by Qt in the future.

---

## Genero Desktop Client 2.2x migration guide

These pages will list differences you may encounter when upgrading to GDC 2.2x.

### Genero Desktop Client 2.21 migration guide

#### Windows: Installer uses MSI technology.

The installer and uninstaller have been rewritten using MSI technology. On Windows 7 or Vista, you will need elevated privileges to run the installer. The exe file we provide asks for elevated privileges, but this is not the case for:

- the .msi inside the .exe. If you manipulate the msi file directly (for silent install, for instance) you need to run it in an elevated command line prompt.
- the uninstaller. To uninstall GDC on Windows 7/Vista properly, use the **Setup** shortcut in the Start Menu and run it as Administrator.

#### ActiveX: embedded mode de-supported

**Note:** Earlier versions of GDC Active X proposed an *embedded* mode (the main window could be directly embedded in the html page instead of the monitor) . Unfortunately, we experience lots of focus conflict in this mode: in Genero, the focus is managed by the runtime system and not by the front-end. In embedded mode, system events can't be filtered as precisely as in *classic* mode, which leads to unacceptable focus issues. Therefore it has been decided to remove this too buggy feature. classic Active X mode is still available and supported.

#### Windows: Default font size is 8.25

While implementing the feature request *support non integer font size*, we noticed that any font copy was using an integer font size value. So this means that the default font size was not 8.25 as the monitor shows, but only 8. This issue is now fixed ; while this is probably not noticeable in most of the cases, this may have an impact if you designed your form exactly for a given resolution.

#### Actions without names are now visible

Bug #14890 - Actions without names are not displayed - has been fixed in 2.21, to match TUI:

```
MAIN
  DEFINE cmd1 STRING
  DEFINE cmd2 STRING
  LET cmd1 = "cmd1"
  MENU "test"
  COMMAND cmd1
```

```

COMMAND cmd2
COMMAND "exit" EXIT MENU
END MENU
END MAIN

```

GDC now behaves like TUI:

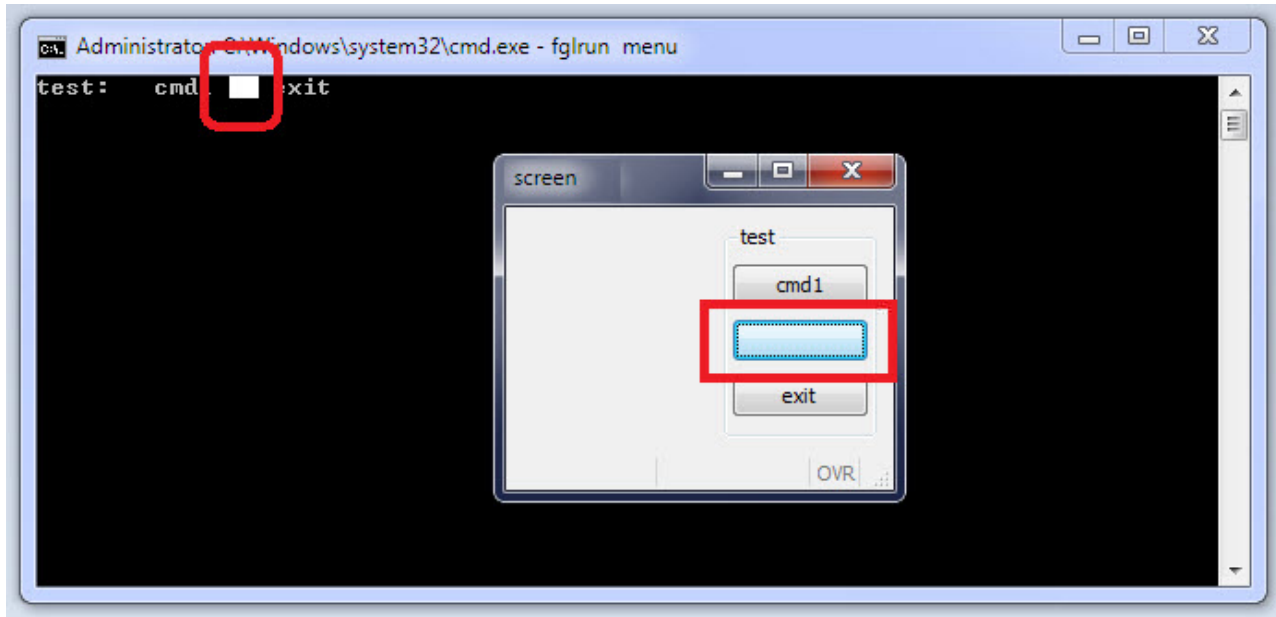


Figure 4-23. GDC behaving like TUI

This may have an impact on your programs if you are using actions without names, which was a classic pattern in early Genero Days:

```

MAIN
DEFINE commands ARRAY[4] OF STRING
DEFINE idx INT
LET commands[1] = "cmd1"
LET commands[2] = "cmd2"
MENU "test"
  BEFORE MENU
    FOR idx = 1 TO 2
      SHOW OPTION commands[idx]
    END FOR
    FOR idx = 3 TO 4
      HIDE OPTION commands[idx]
    END FOR
    COMMAND commands[1]
    COMMAND commands[2]
    COMMAND commands[3]
    COMMAND commands[4]
    COMMAND "exit" EXIT MENU
  END MENU
END MAIN

```

This will result in an extra button with no label on your screen:



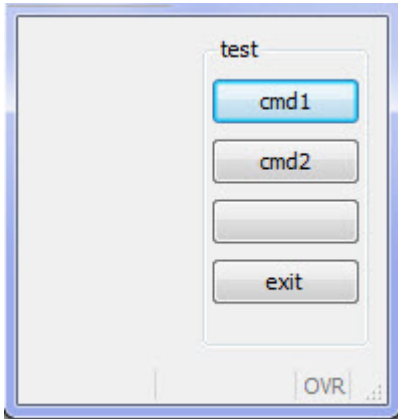


Figure 4-24. Extra button with no label

The reason is that

```
FOR idx = 3 TO 4
  HIDE OPTION commands[idx]
END FOR
```

is actually:

```
HIDE OPTION commands[3]
HIDE OPTION commands[4]
```

which is:

```
HIDE OPTION ""
HIDE OPTION ""
```

that is, hide twice the first action named "". The runtime system has no way to know you would like to hide a different option as they all have the same name. Therefore only the first action without a name is hidden, and all the other are visible.

To solve this issue and get the expected result, you have to give a different name to each of your actions:

```
MAIN
  DEFINE commands ARRAY[4] OF STRING
  DEFINE idx INT
  LET commands[1] = "cmd1"
  LET commands[2] = "cmd2"
  LET commands[3] = "cmd3" --give a name even if not used
  LET commands[4] = "cmd4" --give a name even if not used
```

## Genero Desktop Client 2.20 migration guide

### GDC is now compiled with Microsoft Windows Visual C++ 2008

As with any VC++ application, GDC needs VC Runtime files - basically DLLs Microsoft provides, the equivalent of the glibc on Linux. The difference with GDC 2.1x is that Microsoft changed the way the runtime must be deployed. With VC 2003, it was possible to provide the DLLs with the application, but this was the cause of the DLL Hell. Now The runtime system must be installed on the system directly, using a **VC Redist**. Our installer will always embed the corresponding VCRedist and install it if needed. But, if you were used to simply copying the GDC directory, you have to be sure that the correct redist is installed - if not, GDC will not be able to start.



## Qt4: GUI changes

GDC is based on Qt, a multi platform C++ library. While 2.1x versions (and earlier) are based on Qt3, GDC 2.20 is the first Qt4 based version. Qt4 was a complete rewriting, and in lots of area Qt4 applications are different from Qt3. We've spent a lot of time to minimize the differences, but GDC 2.20 will not be 100% identical to GDC 2.11:

- Some changes are considered to be going in the right direction. For instance, the default font on Windows has changed, but this is to match Windows requirements.
- In some cases, it was technically not possible (or had too high a cost) to work around a change

The behavior of your applications should be unaffected. What may change is the look and feel, and the rendering.

### Default font has changed

In Qt3, there was a bug which made Qt3 based applications not use the Microsoft recommended font ; this bug has been fixed in Qt4, and now GDC uses the font recommended by Microsoft. More details:

- Bug report for Qt3
- MSDN default font information: MS Shell Dlg 2 is the default font for Windows 2000 and after.

You can still set the default font in the GDC configuration panel, if you don't want GDC to use the correct system font.

### Better adaptation to system style

The first version of Qt3 was released in 2001, before Windows XP. The theming mechanism was not designed to make use of all the new features introduced by Vista, OS X 10.4 or KDE4 (and even some items like Folder pages are very poorly supported on XP). Qt4 introduced a much better styling support, relying much more on the system theme - for instance on OSX (and some linux themes) the spacing between items is not fixed and varies depending on the item types. We've adapted these changes to Genero, but for some of them we let Qt apply the system defaults. Windows: Items where the styling mechanism change has an impact:

- Top Menu (and StartMenu as menu) - GDC 2.11 had a Windows 2000 like style for menus. GDC 2.2x is using XP/Vista/Seven style, which is more modern but takes more space.
- Edit based fields now have a 3D effect, a rounded border, and the current fields shows a blue border.
- Comboboxes now have a grey gradient that becomes blue when the mouse moves over it. The side effect is that, in a Display Array, comboboxes on the current rows are not highlighted. (The system theme does not allow changing the gradient color).

### Image attributes are handled differently

GDC 2.20 slightly modifies the handling of images attributes, as there were some inconsistencies in 2.11. Here are the major rules :

- It is important to differentiate the image and the image container (widget): when drawing your form, you're defining an image container. The image(s) that will be put in this container can be smaller or larger.
- SIZEPOLICY and WIDTH-HEIGHT attributes define the size of the container, not the size of the image.
- SIZEPOLICY is the priority attribute. It gives a directive for the size of the image container:
  - FIXED: exact size defined in the Form Specification File.
  - INITIAL: size is computed according to the content of the first display. Once done, the size is frozen. If the content is empty (no image), the container shrinks to its minimum size.
  - DYNAMIC: the width of the container grows and shrinks according to its content.
- WIDTH and HEIGHT attributes define the size of the container, but they are dependant on the SIZEPOLICY. It means the image container may grow or shrink even if WIDTH and HEIGHT are specified. If you really want to have a container with a fixed size, you have to use WIDTH and HEIGHT in combination with SIZEPOLICY=FIXED.
- AUTOSCALE allows the image to be scaled to the space of the image container. It's only useful if size of the image differs from the size of the container. It means there is no interest to use it with SIZEPOLICY=DYNAMIC, as the container always fits to the image size.
- STRETCH allows to make the container grow or shrink (and the image as well) when the parent container (for instance the window) is resized. This attribute doesn't conflict with the others.

#### Examples

```
-- image size: 80*80 pixels
WIDTH=150 PIXELS, HEIGHT=150 PIXELS, SIZEPOLICY=INITIAL, AUTOSCALE;
```

When it is first displayed, the container shrinks to 80\*80 pixels in order to fit the image size. Its size is then frozen. AUTOSCALE is useful here only if another image of a different size is displayed afterwards in the same container.

```
-- image size: 80*80 pixels
WIDTH=150 PIXELS, HEIGHT=150 PIXELS, SIZEPOLICY=FIXED, AUTOSCALE;
```

The container has a fixed size of 150\*150. The image is smaller, and AUTOSCALE allows scaling of the image to the whole space. When not using AUTOSCALE, you may also use the image style attribute *alignment* to define where the picture should be located when the container is bigger.

### Report to printer differences

Introducing Qt4 and Scribe, the text layouting system of Qt has been entirely rewritten. This has an impact on how GDC prints Report to printer: margins, spacings, font size have changed. You may have then to change your report fond to get similar result as before. Genero Report Writer is now the recommended way of producing reports.

### W3C colors

While Qt3 is using X11 color definition, Qt4 is using W3C definition. Some colors have the same name in both definition, but not the same RGB value. This explains why using the term green for a color changed since 2.20.

Table 4-3. RGB values: X11 and W3C comparison

name	X11 RGB value	W3C RGB value
green	#00ff00	#008000
grey	#bebebe	#808080
maroon	#b03060	#800000
purple	#00ff00	#a020f0

## Hitting German ß in an UPSHIFT field results in SS

If you hit the German ß in an UPSHIFT field, it will be immediately replaced by SS. It is something expected: SS is the official capitalization of ß. ß is nearly unique among the letters of the Latin alphabet in that it has no traditional upper case form. This is because it never occurs initially in German text, and traditional German printing never used all-caps.

---

## Genero Desktop Client 2.3x migration guide

### Genero Desktop Client 2.30 migration guide

#### RichText: html generation

GDC is now based on the Qt 4.6 line and we've noticed small changes in HTML produced by textedits:

- `ol` and `ul` (decimal and bullet list) tags now have margin information in style

Before:

```
style="-qt-list-indent: 1;"
```

Now:

```
style="margin-top: 0px; margin-bottom: 0px; margin-left: 0px; margin-right: 0px; -qt-list-indent: 1;"
```

#### Experimental frontcall (re)store size changes

GDC 2.21 introduced experimental front calls to store and restore window size. To follow existing front call names, the two front calls `storeSize` and `restoreSize` have been renamed in lower case: `storesize` and `restoresize`.

#### End of RLOGIN protocol support

RLOGIN is an old and unsecured remote connection protocol. Until now, it was supported for legacy reasons, mainly to have an easy direct `fglty` connection on really old UNIX servers which didn't have any access to a decent SSH server. SSH2 is now the default and recommended protocol for an `fglty` direct connection.

The RLOGIN protocol raises serious issues, the most serious being the need of "root" privileges on UNIX to be able to open a tcp port below 1024. This creates a real security hole, because malicious code could take advantage of `fglty` being launched as root to damage or take control of the system. In addition, more and more UNIX desktop environments (Gnome, KDE, MacOS X, ...) simply forbid such programs to run in a graphical environment without being explicitly "accepted" by the end user (most forbid "sticky bit" completely and display a login box asking for the root password, which nullifies the passwordless login capability of RLOGIN).

Moreover, as with TELNET, RLOGIN doesn't encrypt the communication, so passwords or other sensitive data are transmitted in "clear" channel through the network, another major flaw of the protocol.

For these reasons, it has been decided to remove rlogin support from GDC / fgltty.

## **End of RCP support**

Such as RLOGIN, RCP is now de-supported mainly for security reasons. When RCP was enabled, GDC was allowing any rcp (remote copy) command even if it was not started by a 4GL program. Thus, you were likely to get some unexpected contents. For a similar result, you should rather use FGL\_PUTFILE().

## **Linux: minimum libc is 2.4**

To support fancy GUI features introduced by recent Window Managers like KDE 4, Qt (and therefore GDC) needs to be compiled on a Linux libc 2.4 machine instead of 2.3. As a result:

- If you were running GDC on a Linux libc 2.3; GDC will not start anymore. You'll have to upgrade your system (glibc 2.4 has been released in 2006; we think that for desktop applications like GDC it's better to support recent WM features instead of old systems).
- The GDC theme may change; if you're running a libc 2.4 Linux and a theme which needs libc 2.4 features, GDC 2.2x was not able to load it. Now that GDC uses libc2.4 it will be supported, and the default look and feel will be adapted to your theme.

## **DateEdit: default date change**

GDC 2.30 now supports the INCLUDE attribute for DateEdit. If the field is NULL, the default date of the calendar will be the first date defined by the INCLUDE attribute, if the current date is not included. This prevents opening the calendar with a date you can't select 10 years later than the last accepted date.

## **Table: default different font size taken in account**

By default, some systems are using a different font for table items than for simple form elements. For instance, on Windows 7, while the default font is Ms Shell Dlg, 8.25, Table font is Segoe UI, 9. GDC 2.2x is already using the different fonts, but has two issues:

- if you simply change the font size (e.g. to 12), the font family also changes (it becomes Ms Shell Dlg, 12 while it should be Segoe UI, 12)
- the row height was computed taking the wrong font size in account.

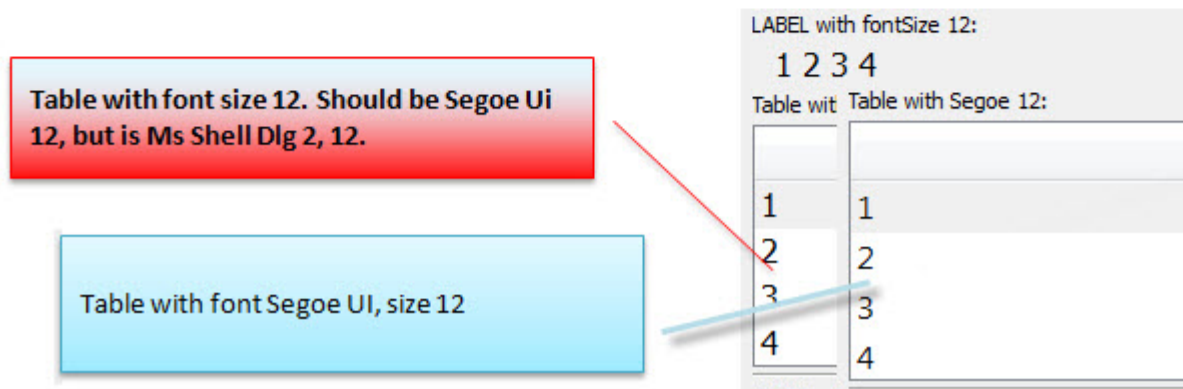
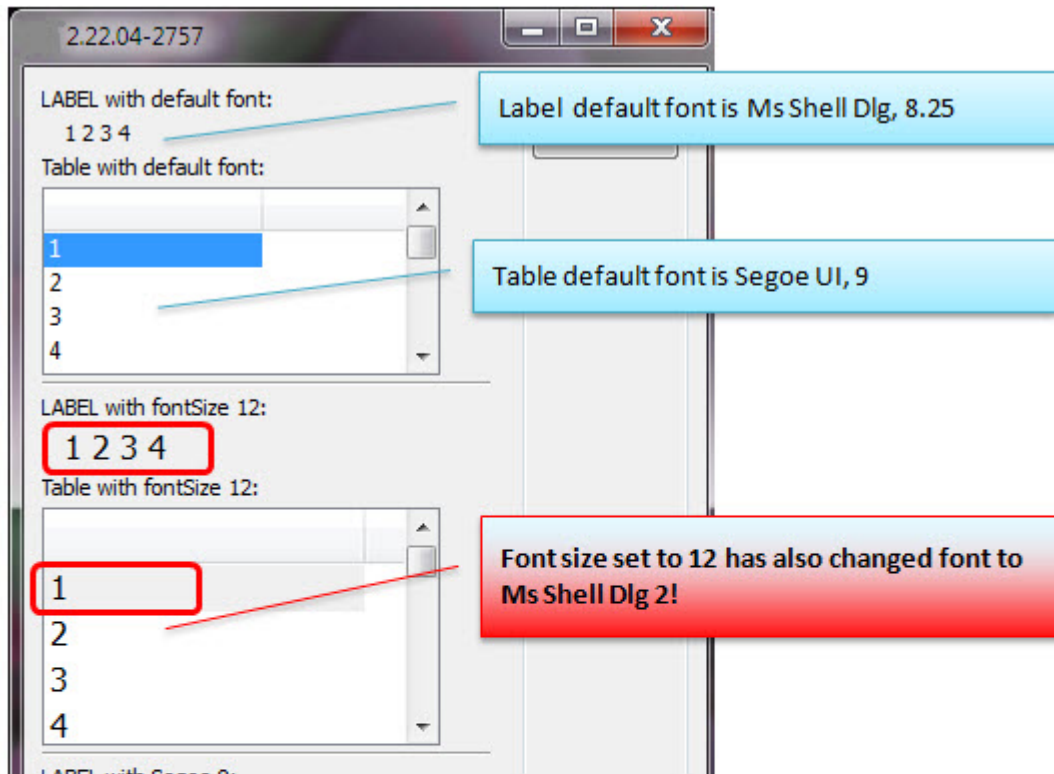


Figure 4-25. Font size examples

These two issues have been fixed in 2.30, but the side effect is that now the table row is taking the right font size, it's height may change depending on the system (from 17 to 18 pixels on Windows 7, for instance).

## Genero Desktop Client 2.4x migration guide

This page will list differences you may encounter when upgrading to GDC 2.4x.

### Genero Desktop Client 2.40 migration guide

This topic lists differences you may encounter when upgrading to GDC 2.4x.

## WinDDE prevents simultaneously connecting twice with the same identifier (program + document)

WinDDE is now preventing from connecting twice simultaneously with a same identifier. This identifier is a concatenation of the *program* and the *document*. For better understanding to what *program* and *document* refer to in WinDDE terms, see the WinDDE documentation. For instance, it may correspond to "EXCEL"+"Sheet1" on Microsoft Excel.

This change has been made because the various DDE objects are stored based on the identifier. When connecting a second time with the same identifier, the previously created DDEObjects were simply lost, as there were no longer references to them. You can no longer call DDEConnect twice with the same identifier without calling DDEFinish(All) in between.

## GDC now listens to localhost only (default behavior)

Before 2.40, GDC opens a network server listening to any connection on a given port (6400 by default). This means that anyone can connect to GDC (depending on your firewall settings). The Security Level mechanism protects your installation, but the tcp port is still open.

GDC 2.40 default behavior is now to listen to the local interface only. Any connection from an other computer will be (by default) denied.

Connection from localhost (for developers having the DVM on the same machine, and when using SSH and port forwarding) will continue working as before. In order to facilitate GDC use for installations not using SSH and port forwarding, GDC will automatically listen to all interfaces when a direct shortcut (except for ssh+portforwarding) is started, therefore existing configurations are still working as before. This "any interface" server will be shut down a couple of minutes after the last connection (application or terminal) is over.

The recommended way to set up a direct connection is to use SSH and automatic port forwarding. In this configuration, your connection is secured by SSH, the communication is encrypted, and as GDC will be listening to the local interface only. No Firewall popup will occur when starting GDC the first time.

Override the default behavior with the command line argument `--listen`.

- `--listen ANY` makes GDC 2.40 work as before, listening to any connection.
- `--listen NONE` prevents GDC 2.40 listening at all. This is suitable when you are running HTTP connections only.
- `--listen LOCAL` makes GDC 2.40 listen to localhost only. This is suitable when you are running SSH + port forwarding and local development only.

The `-D` flag (to activate debug mode) implies the `--listen ANY` mode, to ease development. See GDC command line for more details.

## Modal window behavior change

If you create a window using the `windowType=modal` StyleAttribute, GDC now considers this window as a real modal window:

- the window is modal to a base window (the previous current window)

- the modal window has no entry in the taskbar, and will not appear in the ALT+TAB (or windows KEY+TAB for Windows 7) sequence. Only the base window will appear.

Usually, the "modal chain" must be respected:

- it should be forbidden to close the base window without closing the modal window
- it should be forbidden to make current a non modal window if a modal window is displayed

To ease the migration from earlier versions, GDC will handle these cases by removing the modality of the window.

### **Title of an horizontal menu is now visible**

When setting a title for an horizontal menu (by setting the style attribute `ringMenuPosition` to `top` or to `bottom`), this title is now visible on the form. It was not the case in previous GDC versions. Nevertheless, if the menu doesn't contain any button (because for instance all action views reported are on the `ToolBar`), the title will be automatically hidden.





---

## Chapter 5. ActiveX

These topics cover Genero Desktop Client as an ActiveX.

- "Genero Desktop Client ActiveX"
- "Genero Desktop Client ActiveX and the Genero Application Server"

---

### Genero Desktop Client ActiveX

See also : GDC ActiveX

#### Genero Desktop Client ActiveX Overview

Genero Desktop Client is also an ActiveX.

Packed into its .cab file, it can be put into a web site and then used directly with Internet Explorer.

Once a browser connects to the given URL, Genero Desktop Client will be displayed within a web page and can be used exactly in the same way as the "classic" version.

The first time you access the Client in ActiveX mode, it installs itself and creates a shortcut in Windows Start Menu. Then Genero Desktop Client can be executed from the web page, or directly with the shortcut.

#### Installing Genero Desktop Client ActiveX

If you are running Windows XP Service Pack 2, the installation is described in the topic GDC and Windows XP Service Pack 2.

To install the GDC, connect to the Application Server where GDC ActiveX (GDC/AX) is installed.

Next, you get a certificate prompt. To install the GDC press "Yes".

After a few minutes GDC will be installed on your computer in the default directory.

You will be able to launch the GDC through the Windows start menu (classic GDC) or via Genero Application Server (GDC ActiveX) However it is launched, it can be used in the same way as the normal version.

If you get a new version of the ActiveX, this new version will automatically update your old version.

#### Uninstalling Genero Desktop Client ActiveX

Uninstall the GDC using the control panel, just as you would uninstall any Windows application.

---

### Genero Desktop Client ActiveX and the Genero Application Server

See also : GDC ActiveX

## Genero Desktop Client ActiveX and the Genero Application Server overview

GDC can be associated with Genero Application Server. With this system, you will be able to start a 4GL application simply with an Internet Explorer Browser and a URL.

## Installing Genero Desktop Client ActiveX with the Genero Application Server

We do not provide a separate GDC/AX installation anymore. Each GAS already contains the latest available version of GDC or GDC/AX. Please contact your support center if you need to upgrade GDC/AX without GAS.

## The Genero Desktop Client ActiveX demo

The demo configured by default is gdc-demo, which runs the classic Genero demo browser.

## Genero Desktop Client ActiveX installation issues and solutions

If the message: "[Object not available! Did you forget to build and register the server?]" displays when you start an application, the installation was not completely successful; the new binary has been installed, but it was not correctly registered in the Windows Registry. Versions 1.21.1d and greater implement a system to avoid this problem.

To solve this problem, use the **Control Panel / Add/Remove Programs** utility to uninstall the ActiveX, and start an application again.

If the problem still occurs, the only solution is to remove the ActiveX entries in the registry, by hand:

- HKEY\_CLASSES\_ROOT\CLSID\{2311DF65-9D1A-4DDA-94AA-90568D989633}
- HKEY\_CLASSES\_ROOT\Interface\{A31C5317-4015-4080-BB1B-A6E948E99673}
- HKEY\_CLASSES\_ROOT\Interface\{A90A2B09-D05D-426C-A471-D9FCA74FADA5}
- HKEY\_CLASSES\_ROOT\TypeLib\{0F2D0DFC-EB4B-421A-93BF-DD20351DB07B}
- HKEY\_CLASSES\_ROOT\AppID\{96503F06-661C-4A33-B543-C03F43B89587}
- HKEY\_CLASSES\_ROOT\gdc.MonitorView
- HKEY\_CLASSES\_ROOT\gdc.MonitorView1

**Note:** The ActiveX entries on Windows 7 might slightly differ from Windows XP. For instance the CLSID entry may be:

- HKEY\_CLASSES\_ROOT\Wow6432Node\CLSID\{2311DF65-9D1A-4DDA-94AA-90568D989633}

It is difficult to provide an exhaustive and precise list. The Operating System handles the register itself. It is not handled by the GDC.

## Genero Desktop Client ActiveX template

A template is an html page with some variable items which are expanded by the Genero Application Server. You will find detailed information about customizing a template and configuring GAS in the *Genero Application Server User Guide*.

Starting with 2.30, the template was enhanced to display a small animation while GDC is installed. The template also became a bit more complex; however the basic theory remained the same.

## Template Definition

When you install GDC/AX, a basic template is installed. This template can be modified to fit your needs. The default template consists of two files:

- `fglgdcdefault.html`, in `$GASDIR/tp1`, contains the template itself.
- `fglgdcdefault.js`, in `$GASDIR/web/fjs/ativex`, contains a small JavaScript needed after a modification in Internet Explorer done by Microsoft.

Microsoft has released an update for Internet Explorer that modifies the way ActiveX is handled. (Very) old templates must be changed accordingly. See IE ActiveX update (KB9 12945), Ms Technet about KB 912945, and MSDN about ActiveX activation

The template is composed of several JavaScript functions:

```
function startIt() {
// first activate ActiveX.
gdc = loadGDC("divgdc", "GeneroDesktopClient", "2,40,6,0", "${connector.uri}");
checkGDC();
}
```

This function is called when the HTML body is loaded. This will start the GDC main object. The version given in a parameter is the minimum version required. If the version is too old, ActiveX will update itself automatically.

```
// this function checks if GDC object is ready. If not, wait 1 second and retry.
function checkGDC(){
if(timerID) {
clearTimeout(timerID);
timerID = 0;
}

// gdc is not ready, wait a few seconds...
if (!gdc || !gdc.enabled) {
retryCount++;
if (retryCount <= maxRetry) {
timerID = setTimeout("checkGDC()", 1000);
} else {
if (confirm("Genero Desktop Client could not be started within the
correct time.\nKeep waiting ?")){
retryCount = 0;
timerID = setTimeout("checkGDC()", 1000);
return;
}
document.getElementById("divwait").style.display = 'none';
document.getElementById("divgdc").style.display = 'block';
alert("Genero Desktop Client can't be started, please contact your
system admin");
}
}
else {
//gdc is loaded, hide the loading panel and show gdc panel
document.getElementById("divwait").style.display = 'none';
document.getElementById("divgdc").style.display = 'block';
configureGDC();
return startApplication("${application.id}", "${application.querystring}");
}
}
```

Basically, the GDC object will be ready when the function `gdc.enabled` exists. So the `checkGDC` function checks if `gdc.enabled` exists; if not it waits 1 second and retries. After 30 retries, we give up, something must be wrong.

You can configure the number of retries with `var maxRetry = 30`. Once GDC is ready, we can configure it and start an application:

```
function configureGDC() {

    //uncomment following line to enable Admin Mode
    //gdc.setAdmin(true);

    //uncomment following line to enable Debug Mode
    //gdc.setDebug(true);

    //uncomment and modify the following line if you want to change the
    ping timeout (300 s by default)
    //gdc.setPingTimeOut(120);

    //uncomment and modify the following line if you want to change the
    proxy (using system settings by default)
    //gdc.setProxy("myproxy")

    // this is to ensure that any popup window will appear in front of the browser
    gdc.setFocus();
}
```

You can configure some options which are usually available using the command line, like admin or debug mode. Refer to “Genero Desktop Client ActiveX API” on page 5-5 for details on all available options.

The next step is to start the application.

```
function startApplication(appName, appQueryString) {
    // the serverUrl must be set BEFORE starting the application
    if ("$(connector.uri)" != ""){
        gdc.setSrvUrl(location.protocol + "://" + location.host + "$(connector.uri)" +
            "/wa/r/" + appName + "?" + appQueryString);
    } else {
        gdc.setSrvUrl(location.href);
    }
    gdc.setPictureUrl("${pictures.uri}");
    gdc.setAppName(appName);

    return false;
}
```

The two main functions are `setSrvUrl` and `setAppName`. Their names are slightly inappropriate today, but they are kept for compatibility reasons.

- `setSrvUrl`, which gives GDC the context of the application (complete url, queryString)
- `setAppName`, which indicates to GDC to start a given application. This function really tells GDC to connect to the application server.

```
function preventClose() {
    if ( !gdc ) {
        return;
    }

    // if there still an application running, display a message on close browser.
    if (gdc.applicationCount > 0){
        event.returnValue = "Genero Desktop Client";
    }
}
```

This last function is called by Internet Explorer when the page is closed; a message box indicates that `event.returnValue` (here "Genero Desktop Client") will be stopped if the page is closed, and asks the user to confirm the close. We check that there are applications running before displaying this message - otherwise Internet Explorer can do its job safely.

```
var timerID = 0;
var gdc = 0;
var retryCount = 0;
var maxRetry = 240; //the script will test if GDC is loaded for 240 seconds
```

These few "global" variables are used to test if GDC is ready for 30 seconds. Here you can change `maxRetry` if you want the page to wait more or less time.

## Genero Desktop Client ActiveX API

**Note:** The GDC object can be configured and manipulated via JavaScript calls.

Most of the parameters are likely to be variables, which will be expanded by Genero Application Server. Since these variables will be expanded as simple strings, you need to add quotes, so they will be interpreted as strings in the JavaScript. For example: `"$(pictures.uri)"`

### Mandatory functions

Table 5-1. GDC ActiveX API mandatory functions

Function name	Parameters	Description
<code>setSrvUrl ( url )</code>	<code>url</code> : The url used to start the html page; usually "location.href", or more complex if you are using connector uri, queryString...	Indicates to GDC/AX the URL used. This is needed to correctly initialize the communication between the GDC and the Genero Application Server (GAS).
<code>setAppName ( appName )</code>	<code>appName</code> : Application name as defined in the GAS configuration file; usually <code>"\$(application.id)"</code>	Indicates to GDC/AX that it has to start the given application. <b>Important:</b> Without this parameter, the application will never start; all parameters must have been initialized before calling this function.

### Optional functions

Table 5-2. Optional API functions for GDC ActiveX configuration

Function name	Parameters	Description
<code>setPictureUrl ( url )</code>	<code>url</code> : Default remote path for pictures; usually <code>"\$(pictures.uri)"</code>	Indicates to GDC/AX where to search for images stored on the server side. <b>Important:</b> This requires GDC Version 1.21.1g or greater and GAS Version 1.21.1a or greater.
<code>setAdmin ( mode )</code>	<code>mode</code> : true or false	Defines that GDC starts in Admin Mode (equivalent to -a command line option)

Table 5-2. Optional API functions for GDC ActiveX configuration (continued)

Function name	Parameters	Description
setDebug ( <i>mode</i> )	<i>mode</i> : true or false	Defines that GDC starts in Debug Mode (equivalent to -D command line option)
setPingTimeOut( <i>timeout</i> )	<i>timeOut</i> : a numeric value	Defines GDC ping timeout, by default 120 seconds.
setProxy ( <i>proxyUrl</i> )	<i>proxyUrl</i> : url of the proxy	Defines the http proxy used for the connection. By default, GDC/AX is using Internet Explorer settings; you can define a specific proxy with this method.
setListeningMode( <i>listeningMode</i> )	<i>listeningMode</i> : one of ANY NONE LOCAL AUTO	Allows to configure the tcp server. See --listen command line options for more details.

Table 5-3. Optional API function for returning information

Function name	Parameters	Description
applicationCount	N/A	Returns the count of running applications. Replaces the old getApplicationCount()

---

## Chapter 6. Security

These topics cover security and the Genero Desktop Client.

- “Security levels”
- “GDC and SSH” on page 6-3
- “GDC and SSH simple setup” on page 6-5
- “Port Forwarding and Firewalls” on page 6-6
- “Implementing a Secure Server with GDC” on page 6-22
- “SSH Configuration Troubleshooting” on page 6-38
- “GDC and Windows XP Service Pack 2” on page 6-40
- “GDC and Windows Vista” on page 6-42

---

### Security levels

In previous versions, the Genero Desktop Client accepted all connections that arrived on the listening port, without any verification. In Genero 2.0, the security level was raised to level 2.

You may change the security level with the command line“-A” option, or through the Security tab.

#### Security level 0

Command Line : `gdc -A 0`

Any connection started by the runtime system is authorized without any security message. This security level is the least secure.

#### Security level 1

Command Line : `gdc -A 1`

When the runtime system starts a connection, the GDC user is warned by a Security Connection Message.

#### Security levels 2 and 3

Command Line : `gdc -A 2` or `gdc -A 3`

**Important:** This only works when using a direct shortcut to start an application.

Security level 2 and 3 use a key mechanism:

1. When started, the GDC generates a random key.
2. When a shortcut is started, `@FGL` (and `@FGLxxx`) sets this key into the `_FGLFEID` environment variable.
3. When the `fglrun` process is started, it reads the `_FGLFEID` environment variable and sends the key back to the GDC.
4. The GDC compares the internal key and the key sent back by the runtime system. Only a connection with the identical key is accepted.

If a wrong key connection is encountered:

**Security level 2:** The Security Connection Message displays a message and asks whether to allow the wrong key connection to connect.

**Security level 3:** The connection is simply rejected.

**Important:** If the runtime system you are using does not handle this feature, you won't be able to run an application at this security level.

As of 2.20, the security mechanism also applied to GAS connections. Keys are transmitted by the Application Server to the Runtime System.

## Security Connection Message

Depending on the security level set for the GDC, users may see a Security Connection Message with options to authorize or reject the connection.

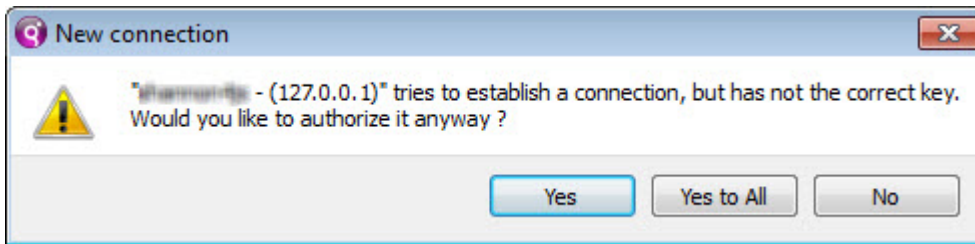


Figure 6-1. Security Connection Message

Table 6-1. Security Message Options

Action	Description
Yes	The GDC accepts this connection and only this connection. The connection information is stored in memory for the duration of the connection. Any additional connection causes the message to be displayed again.
Yes to All	The GDC accepts this connection and any other connection from the same host. This setting is saved to <code>\$GDCDIR/etc/hosts.xml</code> when the GDC closes and is reapplied when the GDC is restarted.  You can modify the <code>hosts.xml</code> file if needed, or remove it to clear the authorized list.
No	The GDC rejects this connection. As a result, the application will not run on the front end.

**Note:** If the user does not have write permissions to the `$GDCDIR/etc/hosts.xml` file, the **Yes to All** option is not available.

Table 6-2. Connection Message and Security Levels

Level	Description
0	Message is never displayed.
1	GDC checks if the IP of the host exists in the <code>\$GDCDIR/etc/hosts.xml</code> file. If not, the message is displayed.



Table 6-2. Connection Message and Security Levels (continued)

Level	Description
2	If the FEID (randomly generated key) is a match, the connection is accepted without displaying the message. Otherwise, the GDC checks if the IP of the host exists in the <code>\$GDCDIR/etc/hosts.xml</code> file. If not, the message is displayed.
3	If the FEID (randomly generated key) is a match, the connection is accepted without displaying the message. Otherwise, the connection is refused outright and the message is not displayed.

The Security Connection Message is displayed depending on the set GDC security level.

## GDC and SSH

### GDC and SSH overview

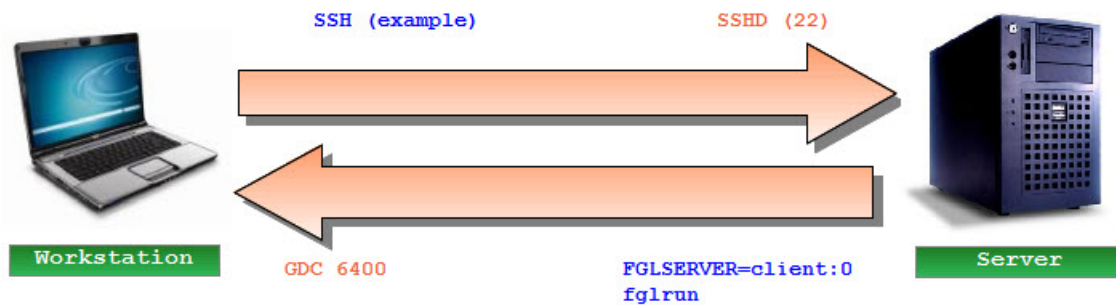


Figure 6-2. SSH communication flow between workstation and server

SSH stands for "Secure SHell". It was designed to replace the 'r' commands like `rlogin` and `rsh`, because they offer no real security. SSH encrypts all data end-to-end, offers data compression, and prevents snooping and connection hijacking. One additional feature it offers is *port forwarding*.

Port forwarding allows an application on one computer to connect to a local port and have its data tunneled through an SSH session to the other computer. This does not require you to open any ports on your firewall router, other than the port you already have open for SSH. This is a distinct advantage. If you have firewalls, this is advantageous as Genero needs to establish a connection from the client to the server to start the user application, and another connection originating from the user application to the client to display the graphical user interface. When Genero establishes a connection from the server to the client, it can use the existing connection to tunnel the graphical connection.

**Important:** Any environment that uses firewalls or connections over the Internet should use SSH or SSH2 for those connections. You should never send unencrypted data such as account numbers, social security numbers, and passwords through the Internet. Some companies might even consider using secure shell connections for internal use when handling sensitive data such as accounting and payroll information. At any point along the way, someone could be monitoring the data, for network diagnostics or possibly with malicious intent. Whatever the reason, encryption is simple and offers peace of mind.

SSH is comprised of two main components, the server component "sshd" and the client component "ssh". Genero provides its own client component (built-in).

## GDC and SSH prerequisites

### Things you should know about your system

In order to determine how to proceed, you will need the following information about your environment:

- Is there a server-side firewall between the server and the client?
- Is there a client-side firewall between the server and the client?
- Is encryption needed for all your data?
- Are you using a VPN (Virtual Private Network) or NAT (Network Address Translation)?
- Will you need protection from inactive sessions timing out?
- Do you have more than one server to access from outside the firewall?
- Do you have more than one client accessing servers outside the firewall?

We recommend reading about SSH and how to configure it. We will not cover this topic in this document.

### How do I make sure data is encrypted?

To ensure that your data is encrypted, select SSH or SSH2. Both offer data compression and port forwarding; the difference is SSH2 has different implementation code and a more advanced encryption algorithm than SSH.

If you are using the shortcut buttons in the Genero Desktop Client, two connections are established between the client and the server. The first connection is established from the client to the server, in order to log in and start the application. The second connection is made from the server's application to the client, in order to display the graphical data.

Use the Table 6-3 to determine which settings you will need.

*Table 6-3. Data encryption selection matrix*

Type of connection	Command encrypted	GUI encrypted
telnet	NO	NO
ssh	X	NO
ssh port forwarding	X	X
ssh2	X	NO
ssh2 port forwarding	X	X

### What connection method should I use?

Knowledge of your configuration will be necessary to make Genero work properly, as discussed at the start of this topic. Use the Table 6-4 on page 6-5 to determine which connection methods will support what you are trying to do. Currently the SSH or SSH2 with Port Forwarding provides the most flexible connectivity.

### Legend

1 - Requires configuring the server side firewall router to open or forward the port used by sshd.

2 - Requires configuring the client side firewall router to open or forward the port(s) used by the GDC.

3 - May require changes to firewall connection timers if firewalls are involved.

X - Indicates full functionality with no changes.

NO - Not supported

Table 6-4. Connection method support matrix

	telnet	SSH	SSH + Port Forwarding	SSH2	SSH2 + Port Forwarding
Firewall or NAT on Server Side	1	1	1	1	1
Firewall or NAT on Client Side	2	2	X	2	X
Firewall or NAT on Both Sides	1,2	1,2	1	1,2	1
Private Network	X	X	X	X	X
VPN (Same as Private Network)	X	X	X	X	X
Encryption of all Data	NO	NO	X	NO	X
Password/login Encrypted	NO	X	X	X	X
Keep Alive	NO	NO	X, 3	NO	X, 3

---

## GDC and SSH simple setup

The simple setup assumes that you are on a corporate LAN with no firewalls. All methods of connections are possible here (telnet, ssh, ssh2, with/without port forwarding) without any special set up. Using SSH or SSH2 will work fine and will offer encryption. The GUI connection will be made on the default port 6400. FGLSERVER will be set to '<client IP> :0' and it will expect to be able to access that IP and port directly.

## Simple connection no Port Forwarding

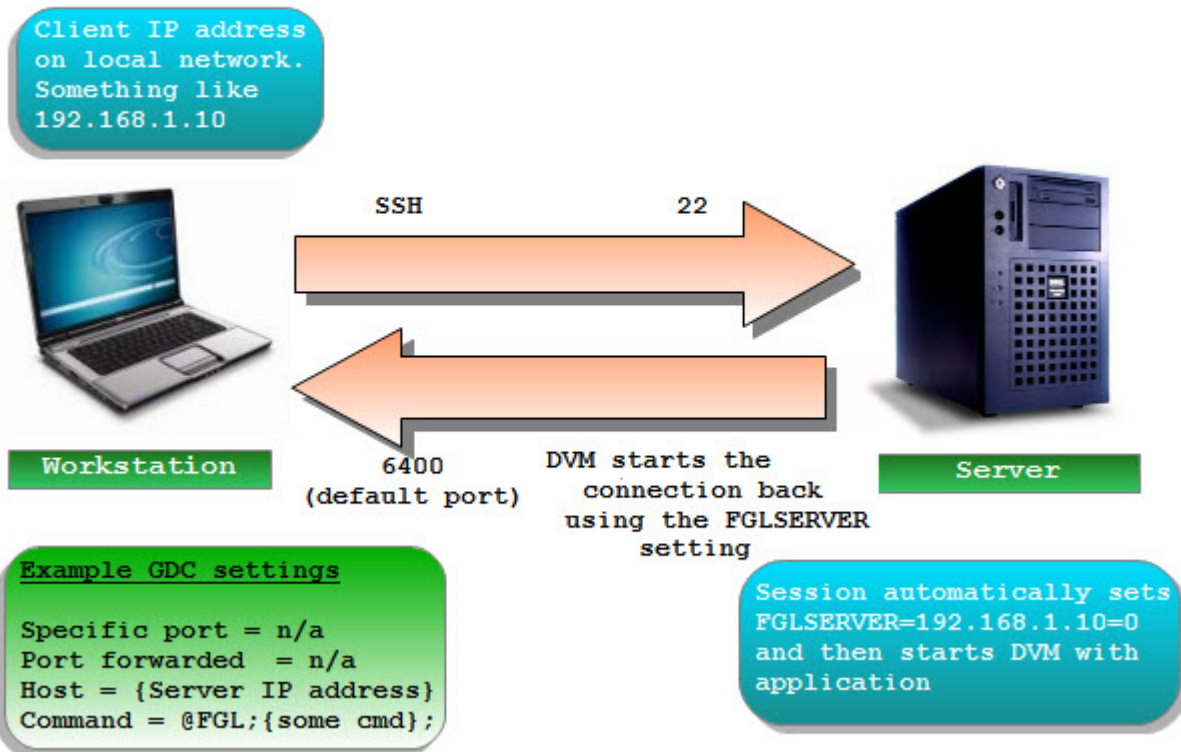


Figure 6-3. Simple connection no Port Forwarding

If you don't want any encryption or compression, select telnet as your method of connection.

What if you want to connect to a port other than 6400 for the GUI? Specify the option "-p <port>" on the command line for GDC, and GDC will listen on that port for the GUI connections. The FGLSERVER will have its information adjusted accordingly. For example, execute "gdc -p 7400". When you look at the value of FGLSERVER, it will contain "<client IP> :1000". It would contain "<client IP> :0" if the default of port 6400 was used (the number displayed after the colon is the port number that you specified minus 6400, the default number.)

If you do port forwarding while using "-p 7400" on the GDC command line, the offset number after the colon will still be your Port Forward value minus 6400. This is because fgllrun doesn't care what port you are listening on the client side, only what port needs to be connected on the server side. The tunnel takes care of connecting to the correct port on the client side. Using @FGL keeps everything automatic. If you have a need for multiple GDC's running at the same time, see Port Forwarding and Firewalls.

---

## Port Forwarding and Firewalls

### Port forwarding

Port Forwarding is used in situations where you want all data encrypted, no session timeouts, or simple firewall setup.

## Simple connection with Port Forwarding

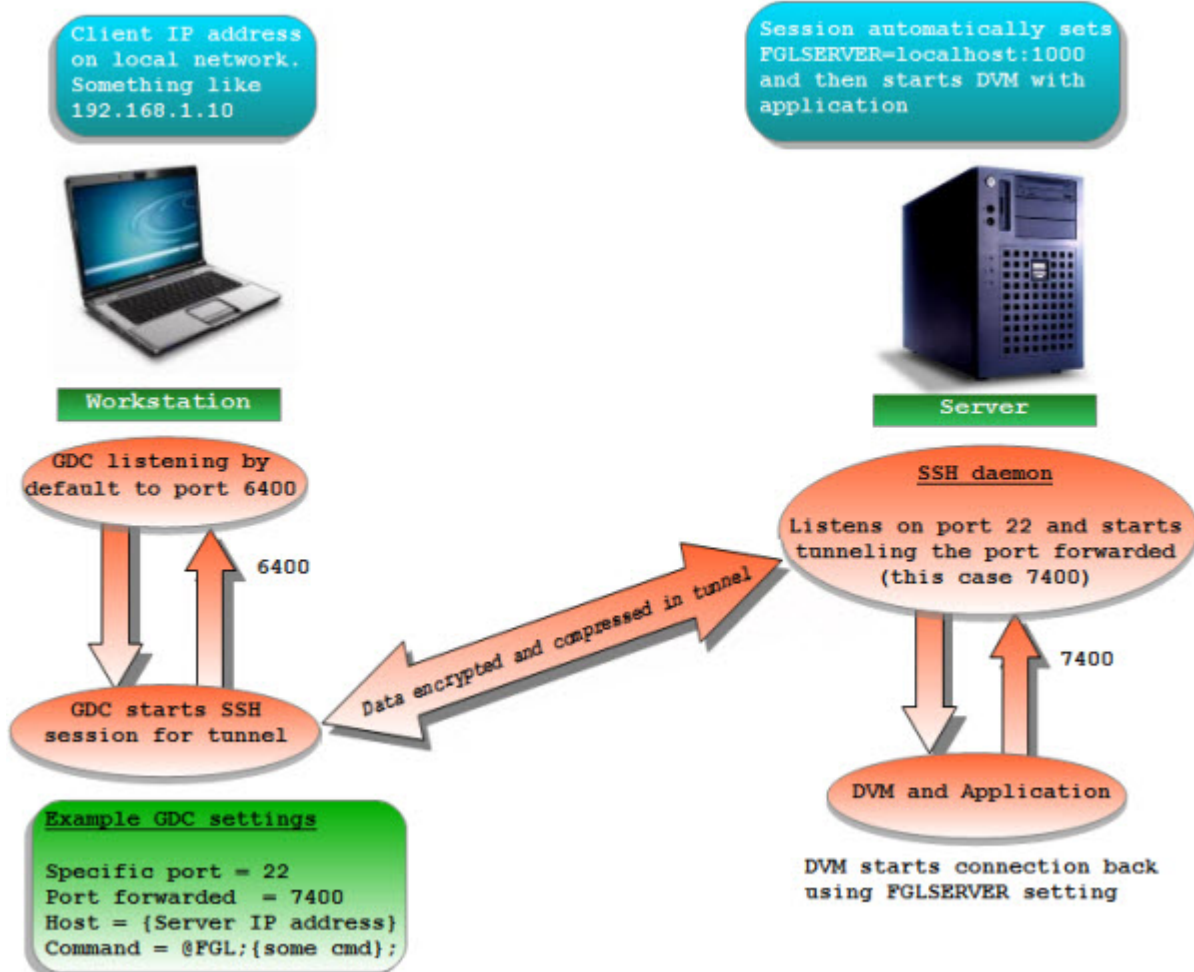


Figure 6-4. Simple connection with Port Forwarding

### Connection to Server side Firewall with Port Forwarding

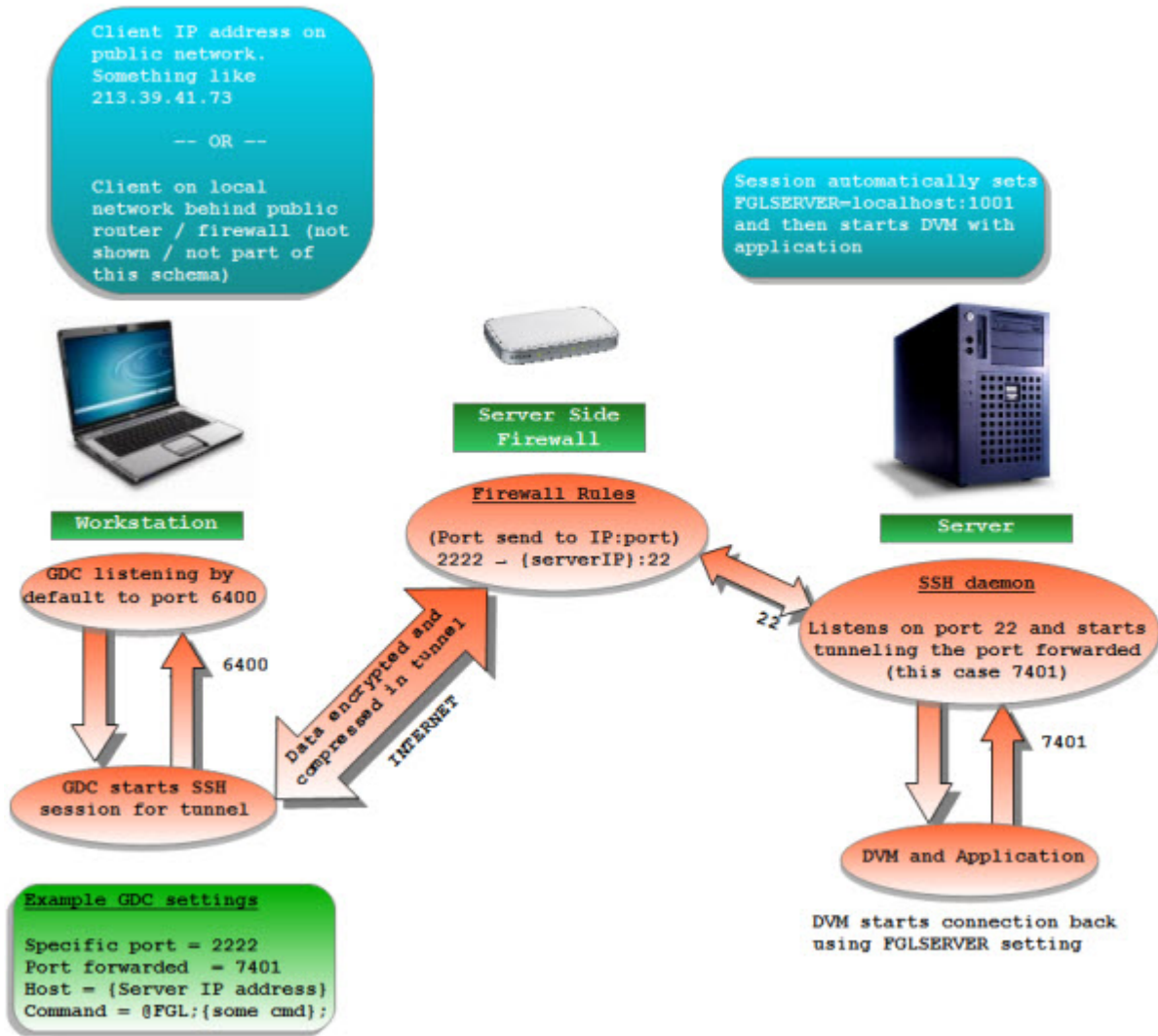


Figure 6-5. Connection to Server side Firewall with Port Forwarding

Figure 6-4 on page 6-7 shows a simple configuration that does not involve a firewall. Sshd, the portion running on the server, will accept a connection from the GDC (client) and start your application. It will also set up a listener for a port that the application will connect to for the GUI. This port is then tunneled through the existing connection to the client, where the client will display the application. Note that both sides still use ports to accomplish this.

You must have ssh installed and set up on the server. If you are expecting to access your Genero application from somewhere on the Internet, you will most likely have a firewall router and must open a port on your router to allow connections to the sshd. See Figure 6-5 for an illustration of this.

Sshd is by default listening on port 22. You can set a port on the firewall to forward to sshd. Whatever port number you use must be set in the GDC using the "Specific Port" field:



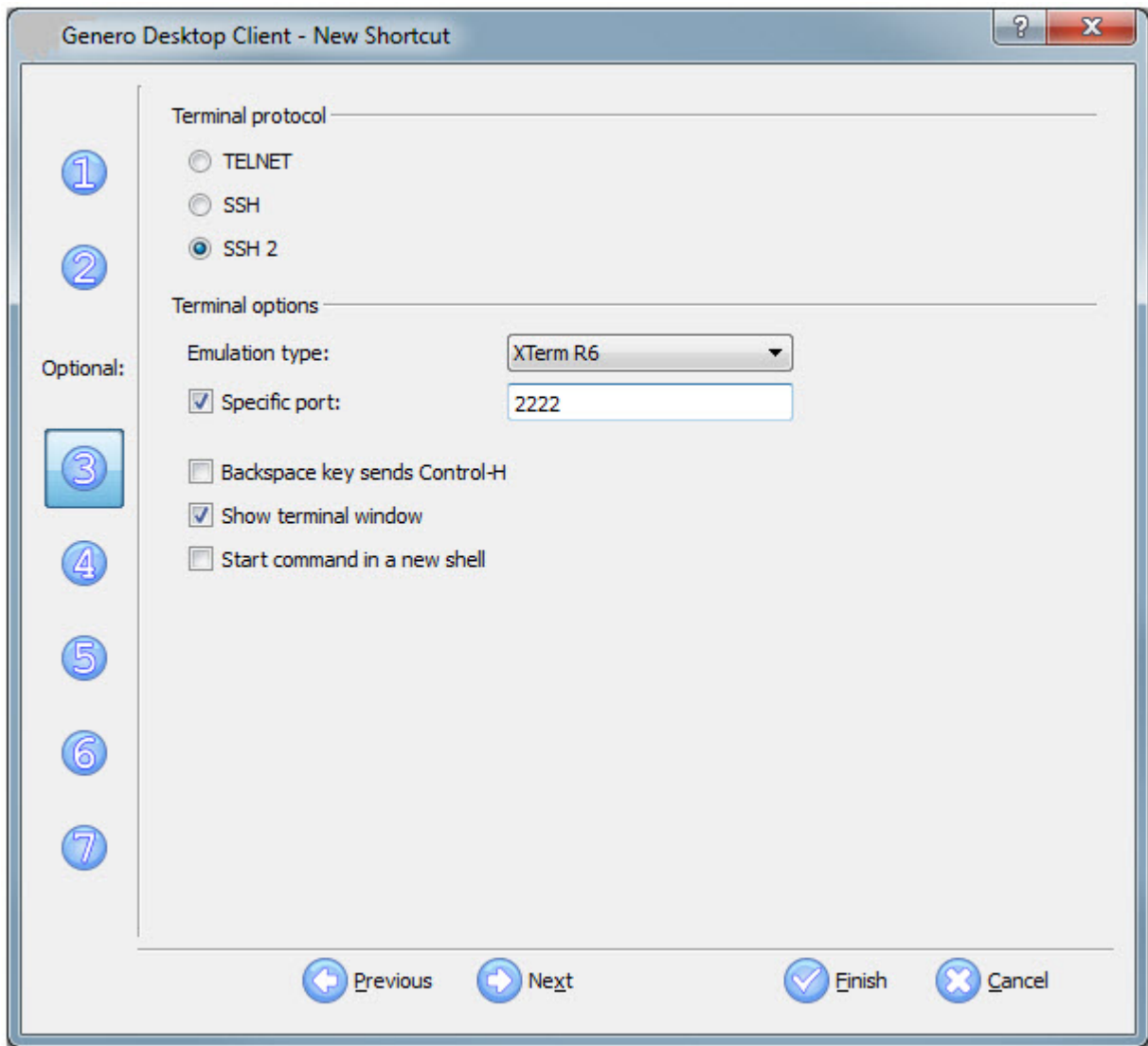


Figure 6-6. Specify specific port number 2222

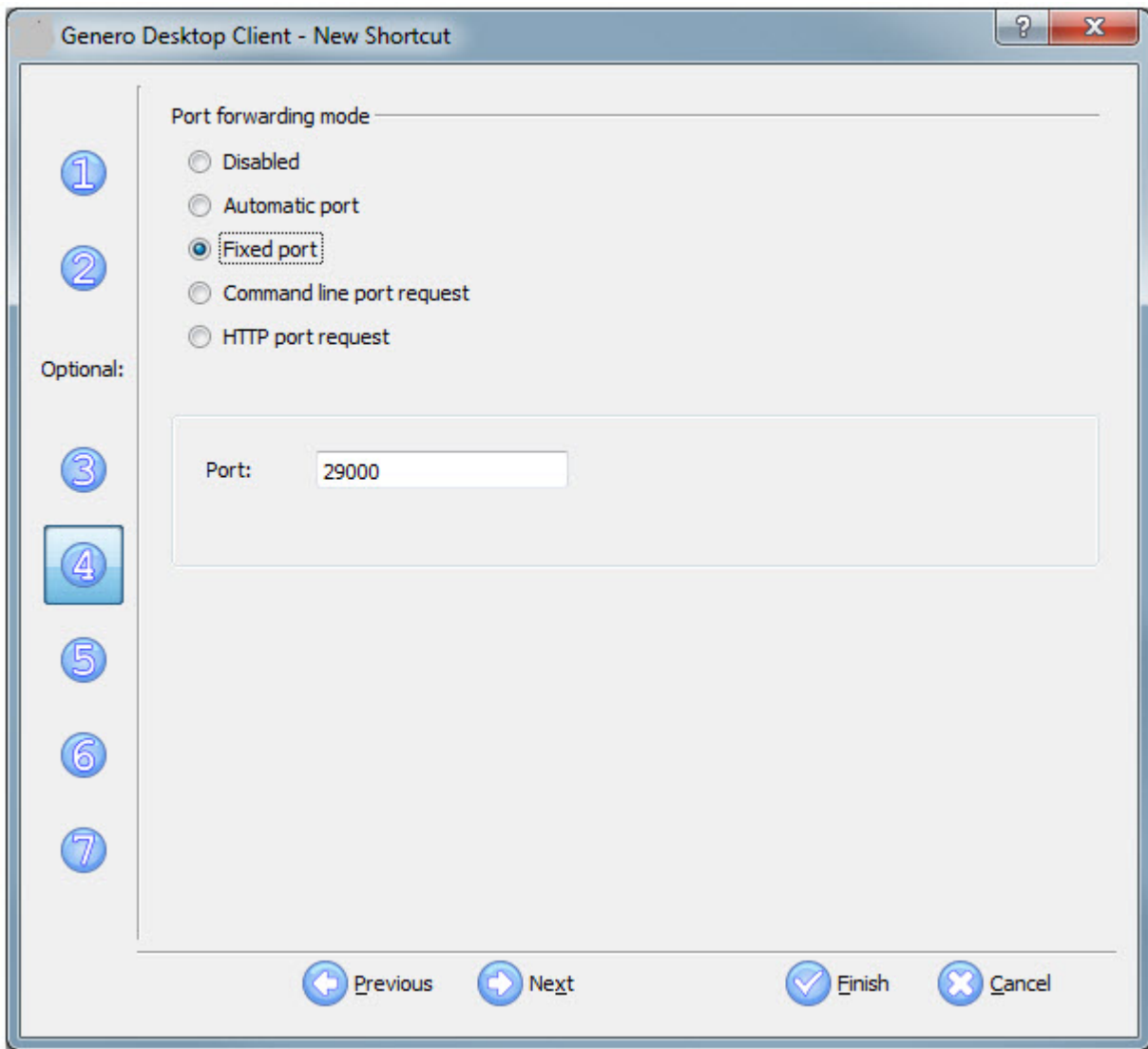


Figure 6-7. Specify fixed port number 29000

In Figure 6-5 on page 6-8 we have set our firewall router to forward port 2222 to our server sshd. There is no reason you couldn't just use port 22 and pass it straight through to your server. If you have more than one server you need to access from outside your firewall, you must use different port numbers and map each server with a different port number. Most routers will allow the destination port to be different from the origination port. For example, a rule could be entered into your firewall router to forward port 2222 to a server on port 22; set another rule to direct 2223 to a different server on port 22, and so on. More details on this are in the Firewall Server Side section.

In Figure 6-7 we have also set Port Forwarding to 29000. This will cause the sshd running on the server to listen to port 29000 for connections from the application. The FGLSERVER environmental variable will be set to 'localhost:22600'. It is localhost because it will be tunneled and sshd is running on the same machine. The 22600 is an offset for the port. To clarify, Genero GDC listens on 6400 by default and any number after the colon in FGLSERVER is added to this number. So 22600+6400 works out to be the port we specified on the client side configuration, 29000.



To use *Automatic Port Forwarding*, you can specify a command line that will execute on the server and return a free port number. As this application is really depending on the system where the Runtime System is installed, we can't provide a version for each system. This program must be used in combination with the GDC connection strings system.

Another way to achieve automatic port forwarding is to have a service running on an HTTP server. This can be a CGI. The program must return lines containing information for the coming SSH connection. One line is always like the following: <attribute name>=<attribute value> For the moment, the attributes managed are "host" and "port", which can indicate the host IP to connect to and the port the sshd will listen to on the server side. By default, the host IP is the same as the HTTP server machine.

Click "Next" for the configuration.

The IP address is that of the server machine unless the firewall on the server side is doing NAT (Network Address Translation). If it is doing NAT, the IP address should be set to the address of the firewall router. Put @FGL on the line labeled "Command Line", so Genero can set the FGLSERVER variable for you when it logs into the server. FGLSERVER will have the port number corresponding to the "Port Forwarding" value you put in the previous screen. Several commands can be placed on the command line and executed in succession. In UNIX you use a semi-colon (;) and in Windows you use two ampersands (&&) to separate the commands.

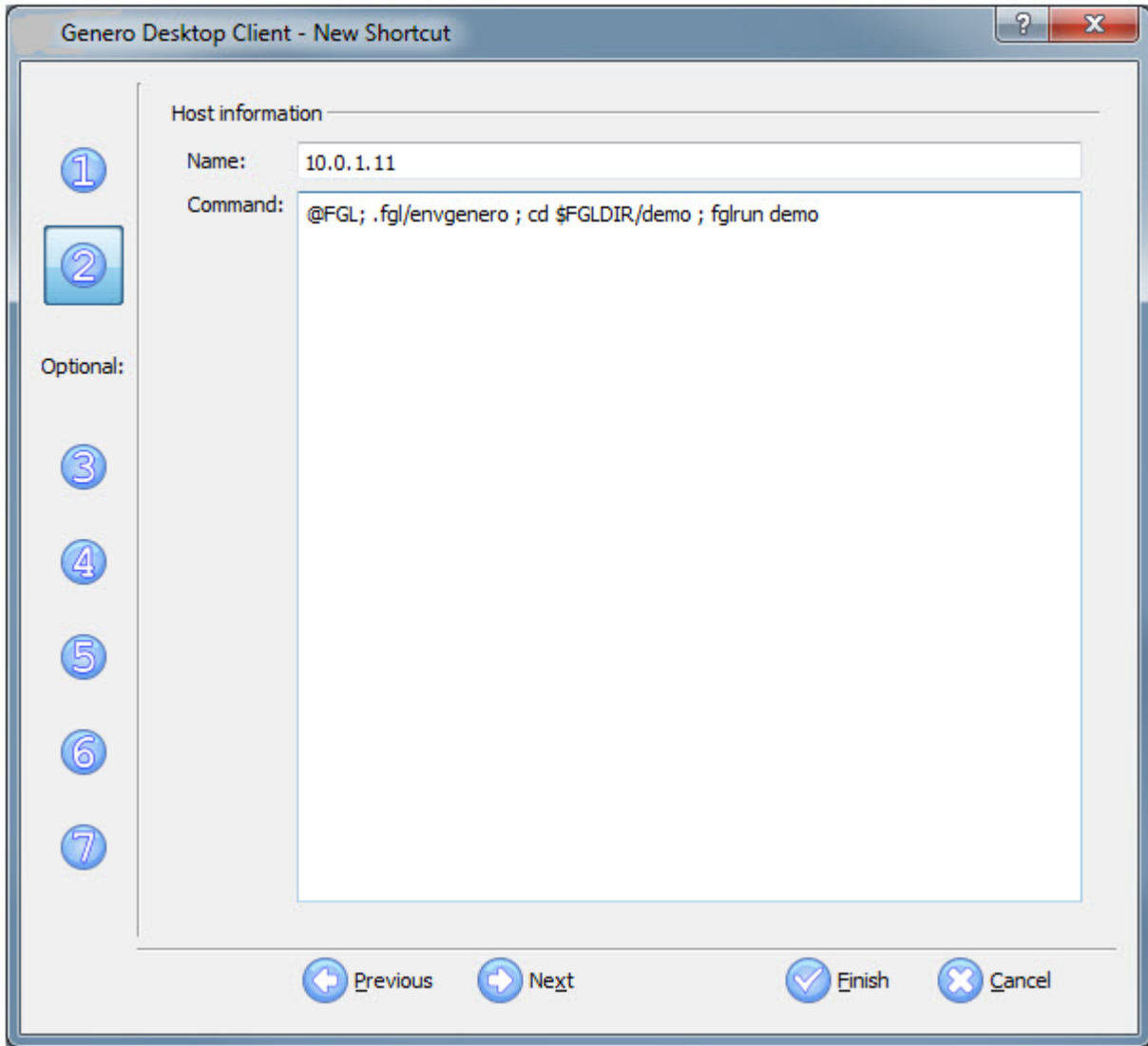


Figure 6-8. @FGL command example

## Port forwarding and the client-side firewall

This section details how to configure the client side.

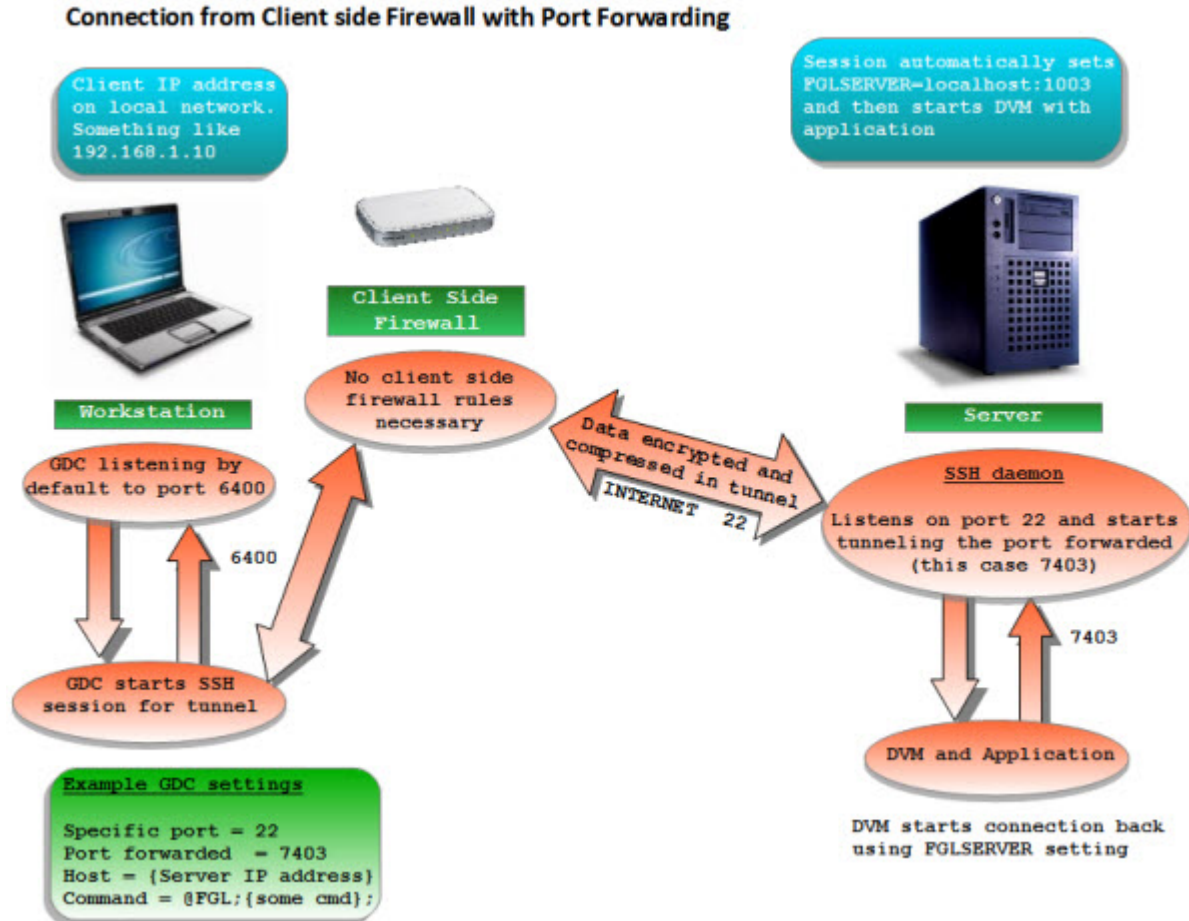


Figure 6-9. Connection from client side firewall with port forwarding

If you have a client side firewall, you cannot connect directly to your clients from outside the firewall. There are two solutions to this problem:

- First, you can set up port forwarding while using SSH or SSH2 (See Figure 6-9). This is by far the easiest and most secure method to connect without the help of a VPN.
- The second method requires adding rules to the router to allow connections (See Figure 6-10 on page 6-14). The set up of the router will be covered here; port forwarding is covered in a separate section.

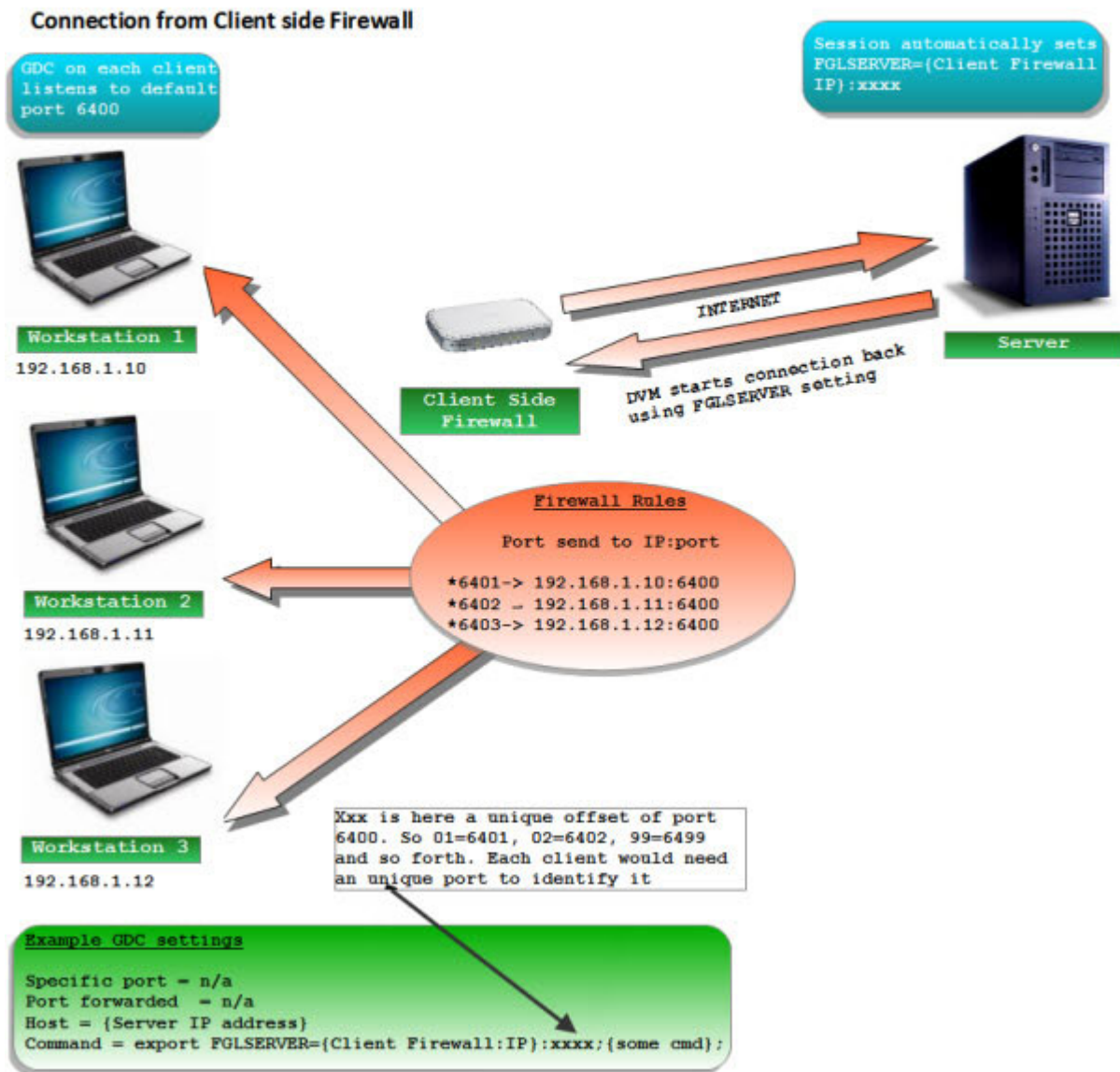


Figure 6-10. Connection from Client side Firewall

The router will need rules added to take a connection coming in on a specific port and direct it to one of your clients. The way Genero is normally configured, all clients would use port 6400. If you only have one client, you can add a rule to the router to forward 6400 to the client on port 6400. If you have more than one client, you will need to allocate other ports on the router to forward to the other clients.

**Note:** In the examples shown, the internal addresses are not public IP addresses. If you have public IP addresses on each client, you can open port 6400 for each of the clients.

Example rule:

Incoming 6400 -> 192.168.1.10:6400

If you have more than one client, you can map them as follows:

```
Incoming 6401 -> 192.168.1.10:6400  
Incoming 6402 -> 192.168.1.11:6400  
Incoming 6403 -> 192.168.1.12:6400
```

Another option if your firewall won't allow you to change the destination port number:

```
Incoming 6401 -> 192.168.1.10:6401  
Incoming 6402 -> 192.168.1.11:6402  
Incoming 6403 -> 192.168.1.12:6403
```

This last example requires that you start the GDC with the `-p` option, causing it to listen on a different port from the default port.

```
>gdc -p 6401  
>gdc -p 6402
```

If you are setting up multiple clients in this manner, you may want to avoid starting the first client on 6400; any misconfigured new clients will pop up on that user's console unexpectedly.

On the command line of the GDC shortcut setup, assign `FGLSERVER` to be the IP of the firewall router with the corresponding port of the router. This must be hard-coded, since there is no way for the client computer or Genero to know how the connection is established.

For example, if the client firewall router's IP address to the Internet is 213.39.41.73, and port 10000 is mapped to the client 192.168.0.53 port 6400, then the entry in the router would be:

```
Incoming 213.39.41.73:10000 -> 192.168.0.53:6400
```

The command line in the GDC would look like:

```
FGLSERVER=213.39.41.73:36000; fg1run demo
```

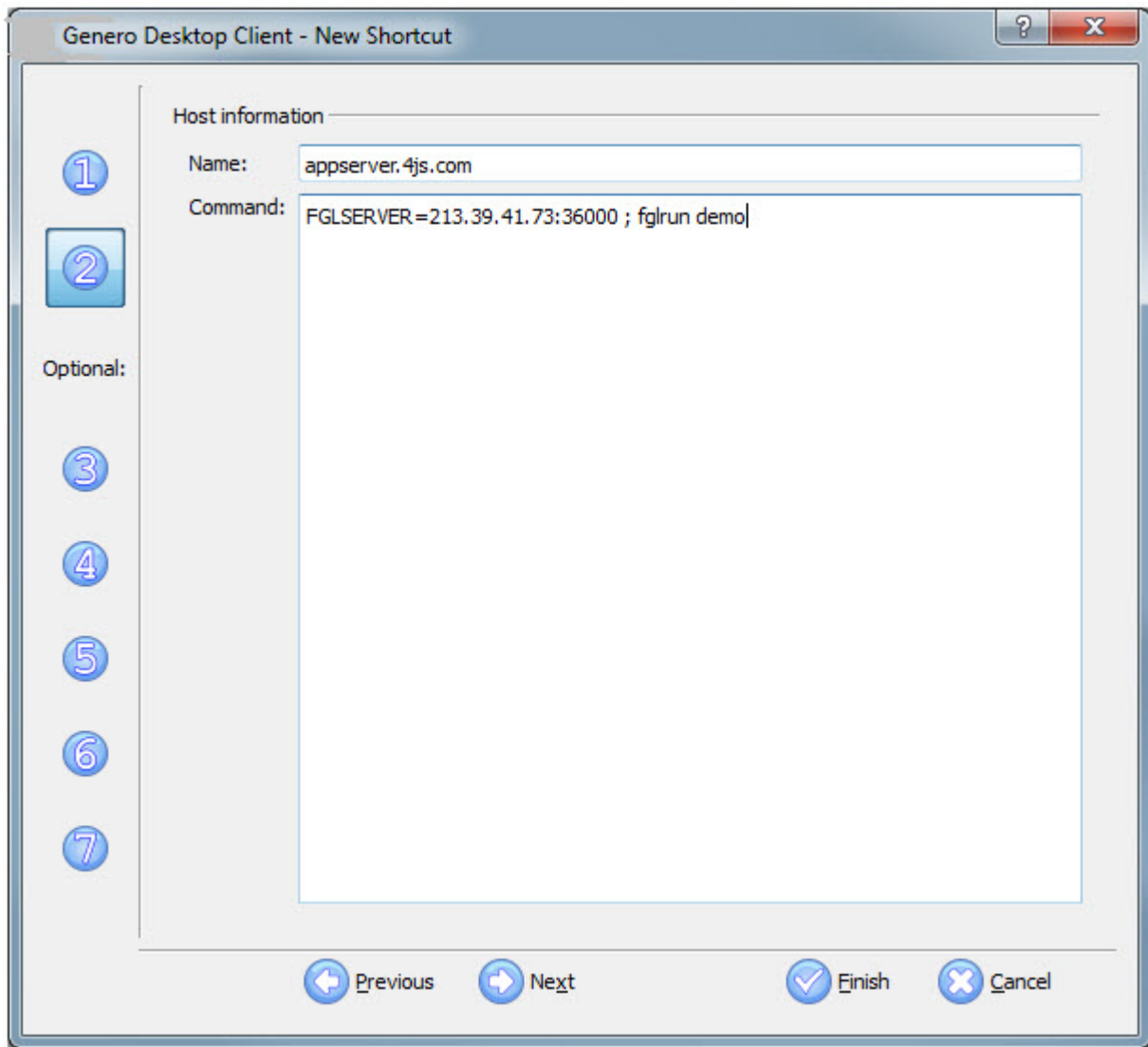


Figure 6-11. Entering the proper command for a GDC shortcut

The FGLSERVER variable is normally set using @FGL, but that would set FGLSERVER to the IP of the local client machine and the port specified when the GDC was started with -p. If the IP addresses used behind the firewall are public, this would be OK. If the addresses are not public, however, we must use the IP address of the router, and let the router translate and forward it. If the router is translating the port, then we must use the port that the router is expecting.

In our example the port that the router is looking for is 10000. The FGLSERVER port value must be set to 10000 minus 6400, resulting in 3600. This is because FGLSERVER=<ip> :0 tells Genero to connect on port 6400. The number after the colon is added to 6400.

## Port forwarding and the server-side firewall

Having a server side firewall is the typical configuration on many systems. There is only one method for doing this, whether you use telnet or ssh: map a port to be forwarded to the server in the firewall router. It is not advised that you use telnet from the Internet for security reasons; that is usually why you have a firewall.

Decide which method of connectivity will be allowed, and determine what port you will use to forward to this service. If there is only one server involved, you can use port 22 for ssh or 23 for telnet and forward them straight through to the server. But if there are several servers involved and they do not have public IP addresses, you will need to pick different ports on the firewall router and let the router forward those ports to the different internal servers.

See Figure 6-12 for an example of how to do this for a telnet connection. Notice that the returning GUI path doesn't require any special handling unless there is a client side firewall. For details on this see the Client Firewall section.

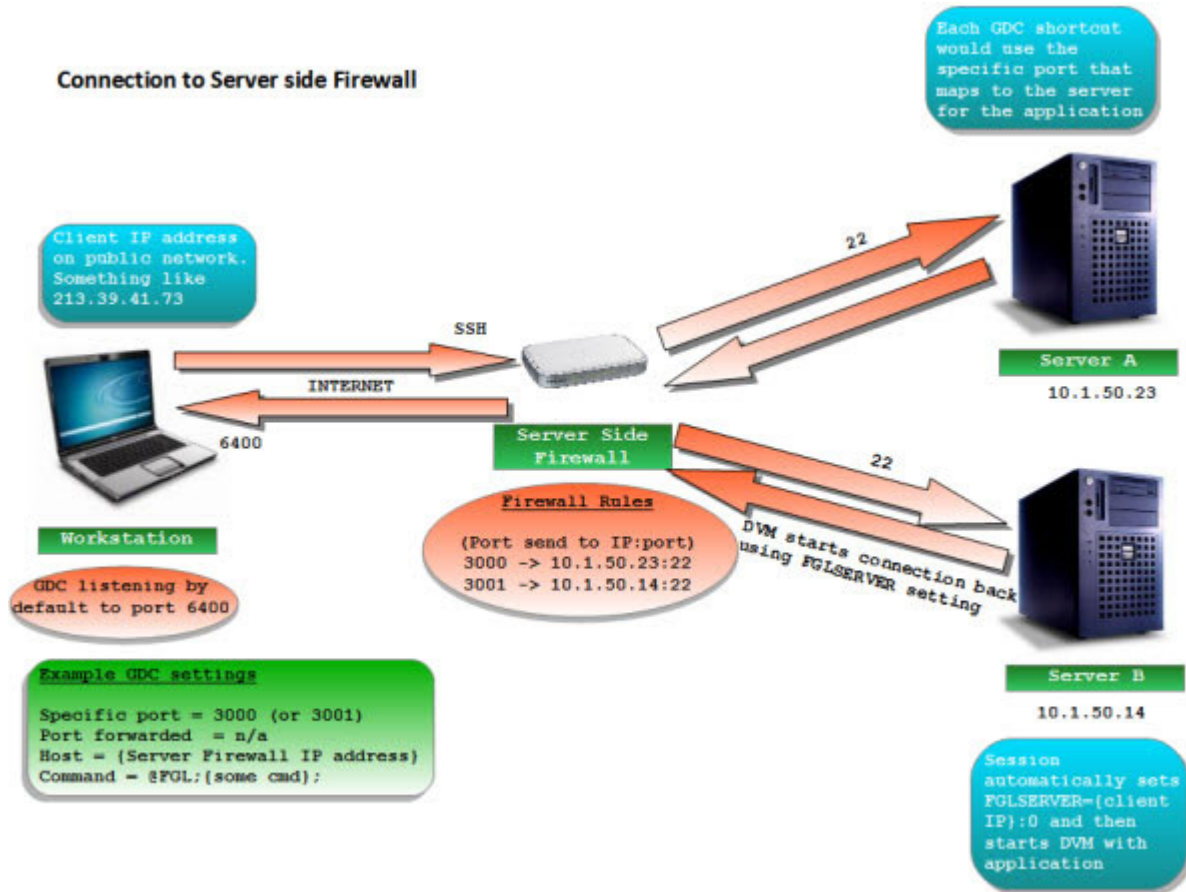


Figure 6-12. Connection to server side firewall

See Figure 6-13 on page 6-18 for an example of how to do this using ssh with port forwarding.



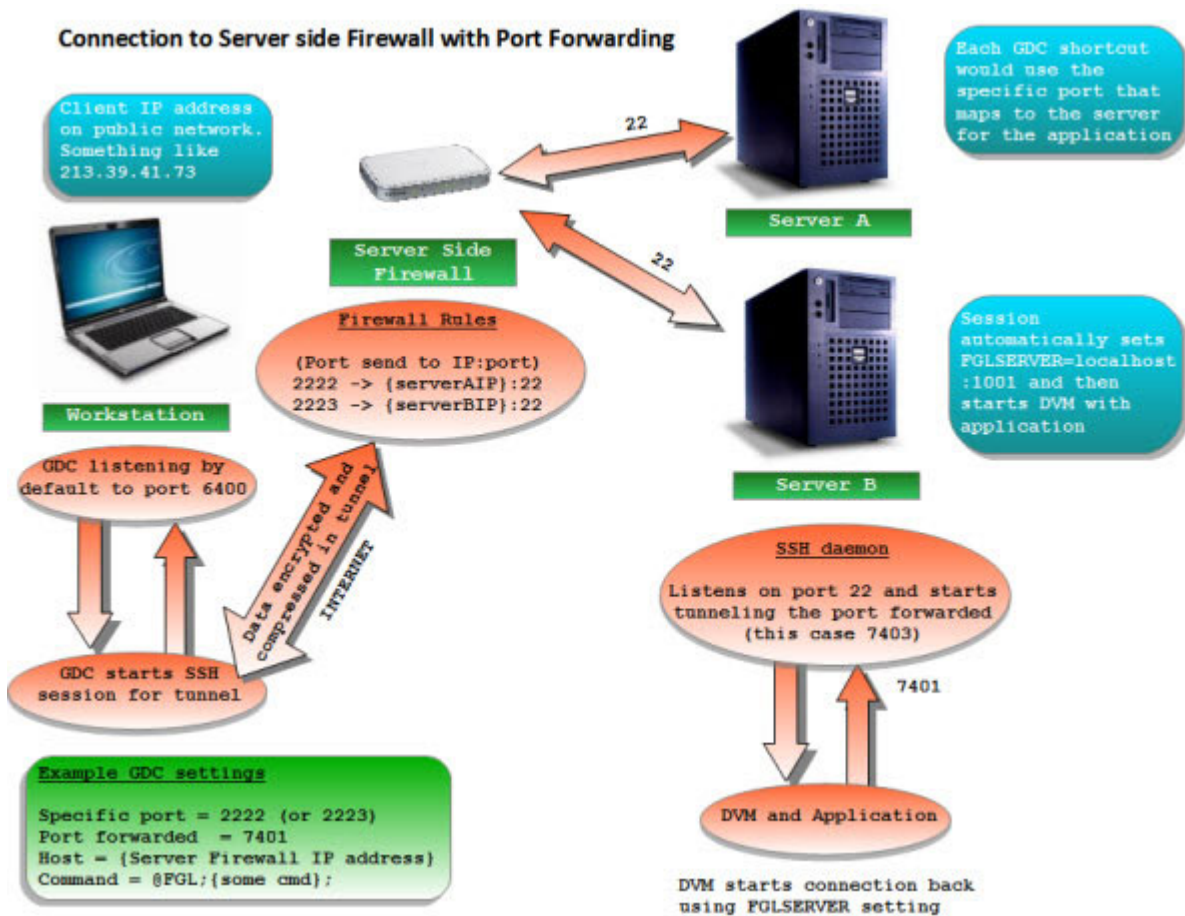


Figure 6-13. Connection to Server side firewall with port forwarding

The client GDC would connect to the server firewall router on port 3000 to access server 1, and port 3001 for server 2. We chose these ports arbitrarily; almost any port could be used. Numbers below 1024 are reserved for well-known services, so choose numbers above 1024.

Using port forwarding will work without modification because the GUI interface is tunneled through the initial connection, and the port it tells the server application to use is a local port to the server. Of course, the same methods as above must be used if there is more than one server.

Using telnet or non-port forwarded ssh will work also, because connections for the GUI originating from behind the server firewall will be allowed out without special mapping. If there is a client side firewall as well, see client side firewall configuration.

**Example:**

We have two servers that will be accessed via clients somewhere on the Internet. They will use ssh2 with port forwarding to simplify client set up and keep things secure. The firewall on the server side has an IP address of 192.168.50.2 (only valid for this example). We have mapped the two servers:

```
213.39.41.73:3000 -> 10.1.50.23:22 213.39.41.73:3001 -> 10.1.50.14:22
```



The GDC client will need to be configured as well:

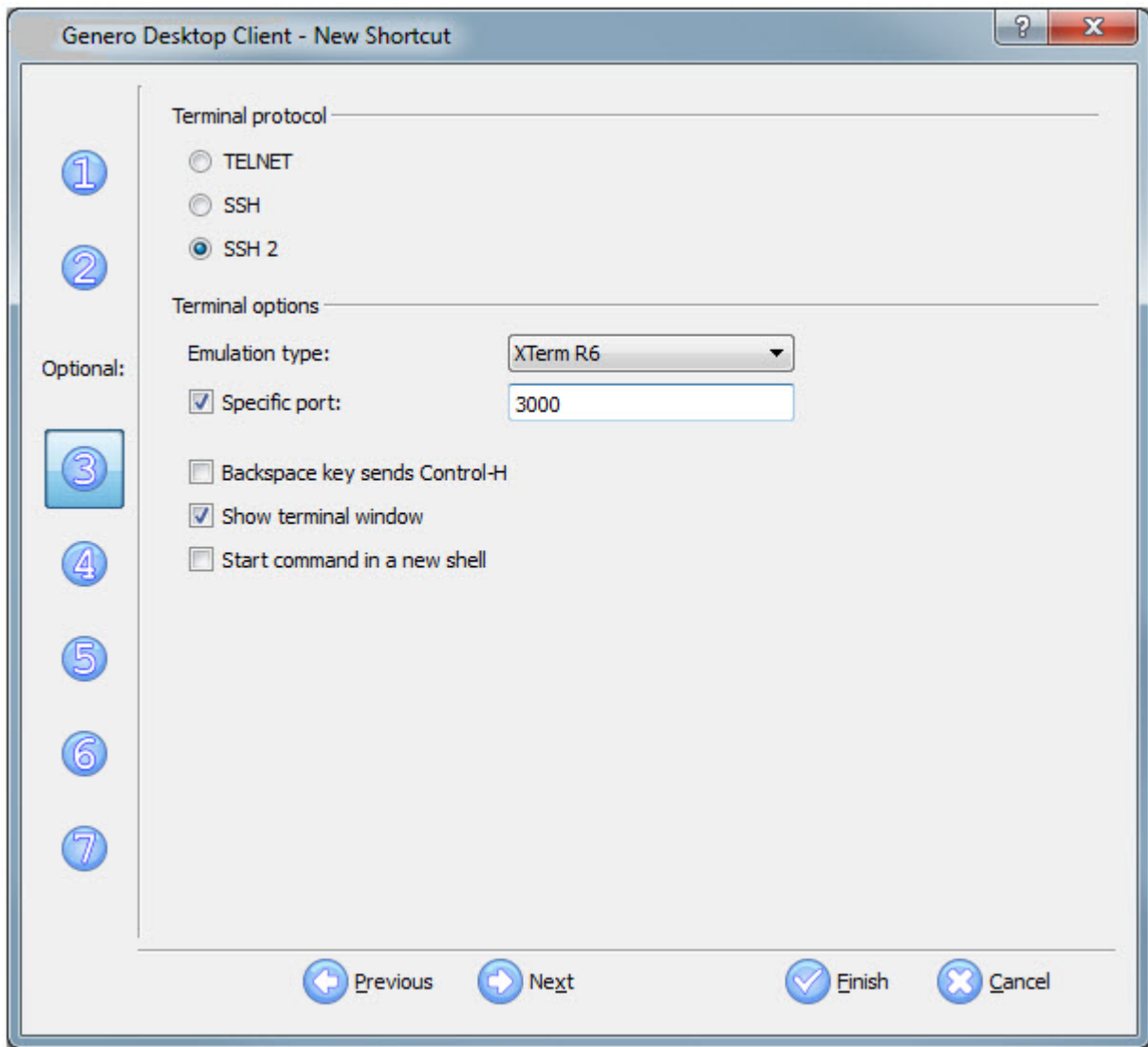


Figure 6-14. Showing configuration for access to Server 1

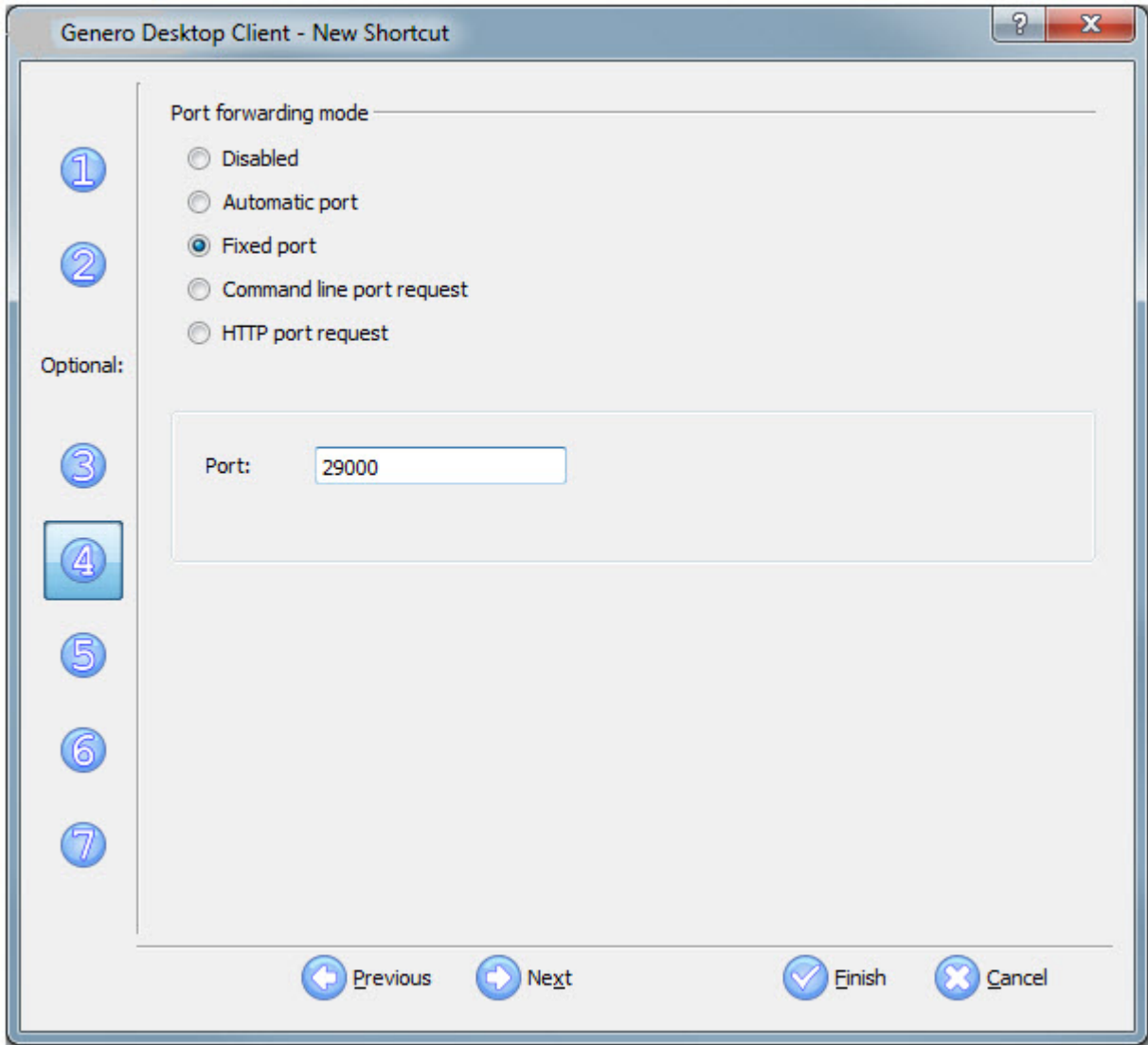


Figure 6-15. Showing configuration for access to Server 1

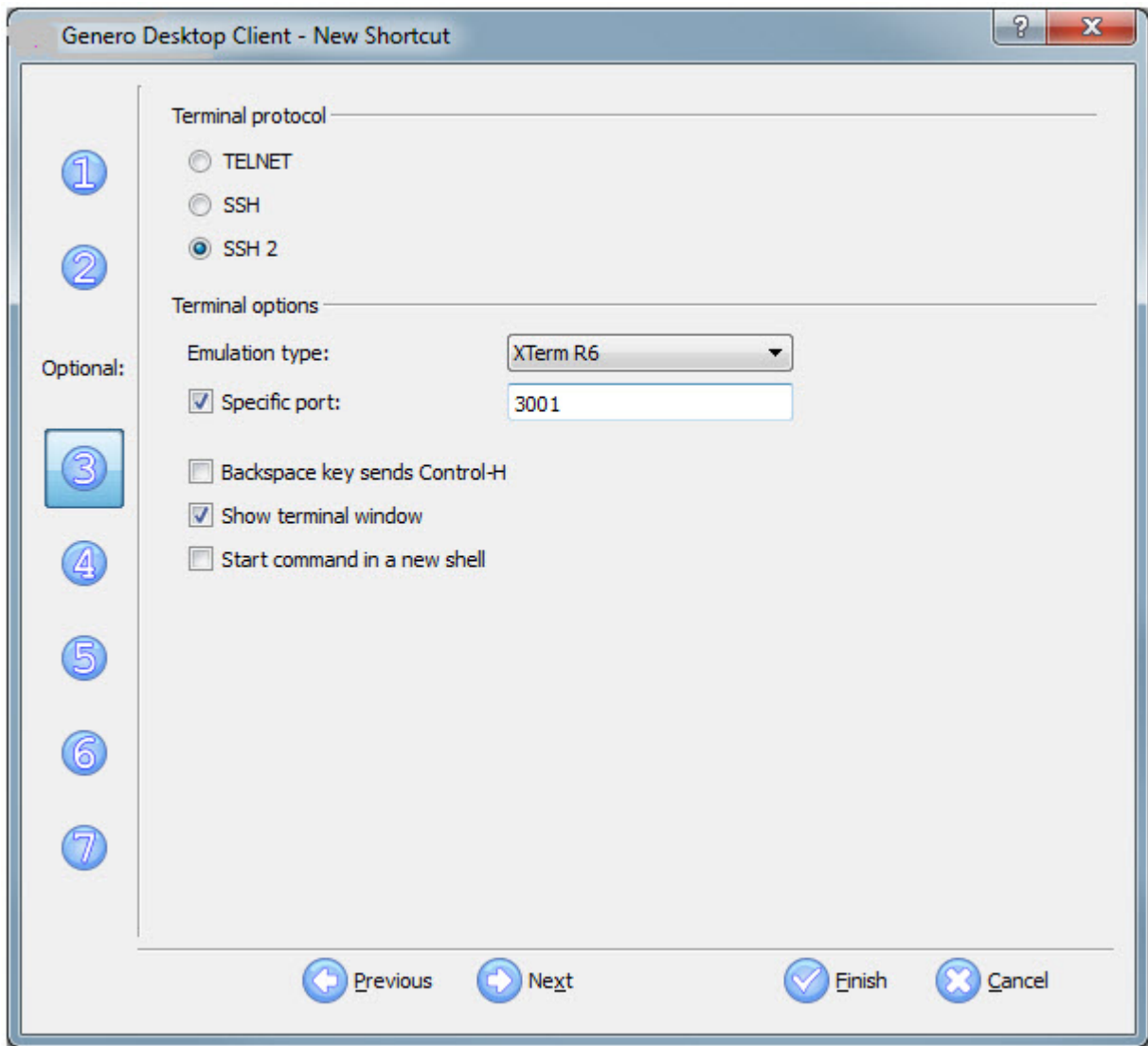


Figure 6-16. Showing configuration for access to Server 2

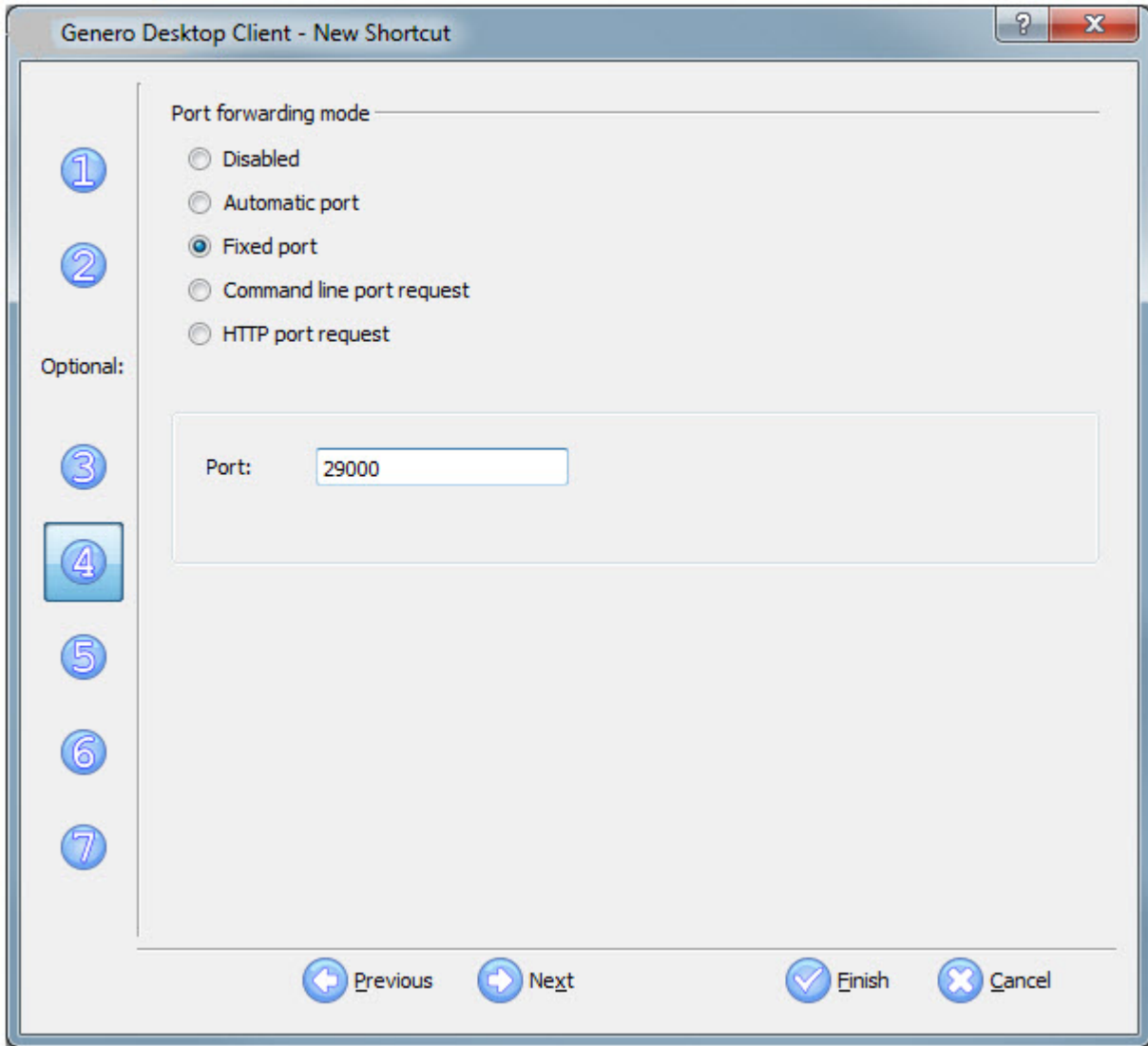


Figure 6-17. Showing configuration for access to Server 2

Figure 6-14 on page 6-19, Figure 6-15 on page 6-20, Figure 6-16 on page 6-21 and Figure 6-17 show how to access each server by specifying the appropriate port for each, one with 3000, the other 3001. This will allow the firewall router on the server side to direct each to the appropriate server. The IP address used would be the IP of the router.

Keep in mind that if you have two users accessing the same server, you must manually select a different port forward number to keep them unique. See Possible Configuration Problems.

## Implementing a Secure Server with GDC

In an enterprise deployment, it is typical for the Genero Desktop Client to be configured to launch in the default user mode with all application shortcuts pre-defined.

When the "-a" or "--admin" option is specified, however, the Genero Desktop Client launches in admin mode, and the user is able to modify existing shortcuts or create

new shortcuts of their own. Therefore, when in admin mode, a Genero Desktop Client user with sufficient knowledge can modify the string passed to the server (UNIX or Linux) and effectively execute any command. While this is expected behavior -- if they can log in to the server, they can enter commands -- this ability can present a problem in some environments.

The following paragraphs explain how to implement a secure server preventing Genero Desktop Client users from executing arbitrary commands, by preventing client access to the (UNIX or Linux) command line or shell while still allowing Genero applications to be started. This is accomplished by not giving them access to the shell, yet allowing the Genero Desktop Client to pass values to the system to indicate which application to start.

**Important:** This is intended to be the framework for a larger implementation and should be reviewed by your system administrator for any security concerns.

## Prerequisites

To implement a secure server, the following prerequisites must be met:

- Genero Desktop Client, version 1.32.1f or greater
- UNIX or Linux platform
- SSH configured on the server
- Familiarity with Bourne or Korn shell programming
- Access to root for implementation

## Solutions overview

When users log in, the system determines which shell to give them, based on a value in the `/etc/passwd` file. We will replace this shell with a shell script that will parse the values passed to it and set the environment accordingly. The application that is started will be from a list of valid applications; no other options will be accepted (thus controlling what a user can do).

### Passing Values to the Script

The Genero Desktop Client must pass specific information to the script:

- The *application name* must be passed if more than one application exists. You can add additional logic to the script to control which users have access to specific applications.
- The *port* accepting connections for the Genero Desktop Client is important so that the application can connect back to the Genero Desktop Client to display information.
- The *two security values* prevent anyone from spoofing the connection. The DVM must make a socket connection to the Genero Desktop Client for the application screens and user interaction. `@FEID` and `@FEID2` contain a value that must match on both the client and server. The Genero Desktop Client compares the `@FEID` value it has internally and the one it received from the DVM attempting to connect. If they do not match, it assumes an application it did not start is trying to connect and rejects the connection. Likewise, `@FEID2` contains a value that the DVM must receive from the Genero Desktop Client in order to validate that the Genero Desktop Client is the one that started it. These security values are enabled by specifying `'-A 3'` as a command-line argument when starting the Genero Desktop Client.

## Auto Port Forwarding

With version 1.30, the automatic assignment of the port to use for port forwarding was added to the feature set of the Genero Desktop Client. Port Forwarding is the term used for tunneling with ssh. It allows applications to connect back to the client via a port that is open on the server, tunneled through the ssh secure client connection, then connects to the Genero Desktop Client on the client. The port is specified by the client, but it is usually not known whether this port is in use on the server prior to initiating the connection. In an enterprise this could be a problem, because every forwarded port must be unique between users.

The solution is to ask the server system for a port number to use. Because there is no way to reserve the port, we must get the number and open it quickly. Once we have the port opened for our session, we will have it until we log off and the connection is closed. We use a small C program that uses network system calls to allow the server to assign a port number. This port number is produced by the operating system by incrementing some internal OS counter and issuing numbers from a pool. If the port it would assign is in use, it will automatically increment the value until it finds an unused port. The next number it assigns to us, or to any other network request, will be managed the same way. This process insures to a large degree that the number we get will not be reassigned or used for some time, certainly long enough for our purposes.

**Note:** Version 2.30 introduces Automatic Port Forwarding. Fgltty is now able to get a free port and pass it to GDC so the ssh tunnel can be set up automatically. In most of the cases this should work and fit your needs, but if you want to assign a specific port number or have a full control over the ports that are used, you can still follow this process:

### Process Summary

- Log in.
- Get a port number from the system.
- Close the connection.
- Establish another connection and provide that port number for the tunnel.
- Log in (again).
- Start the application.

In normal situations the terminal activity of this process is hidden. The users simply see their application appear.

## The shell script

The shell script accepts the information on the command line and parses it, assigning values as needed to start the application. The application name is matched in a case statement, preventing direct execution of what the user sends.

The script provided later in this section is intended to be an example, and we expect you to tailor it according to your needs. Save it in a location where it can be executed but not changed by your users. Edit the `/etc/passwd` file to make a user call the script instead of a shell. Here is an example of the user "user1" running the script named "gdcstart".

```
user1:x:569:569:./home/user1:/home/user1/gdcstart
```

The script `LOGIN_SCRIPT` is designed to recognize the difference between being started from `sshd` or from `telnetd`. You could modify it to handle either condition

differently. For example, you may want it to start an application in text mode when accessed via telnet, or in GUI mode when accessed via ssh.

## Setup SSH login

An advantage of using ssh and port forwarding is that the GUI information is encrypted during transmission. However, the unused port must be assigned on the server for the tunnel -- a difficult task if you are the system administrator. To solve this, we ask the server to tell us what port to use. This section shows how to implement this solution while maintaining system security.

As stated previously, we use a shell script to start the requested application instead of giving the user a shell; the login script is used for that purpose. In order for the script to work properly, the information in the Command Line field of the Genero Desktop Client shortcut must be altered accordingly to launch the application. The automatic assignment of the port forward number must also be set up.

This is the Genero Desktop Client shortcut entry for using ssh.

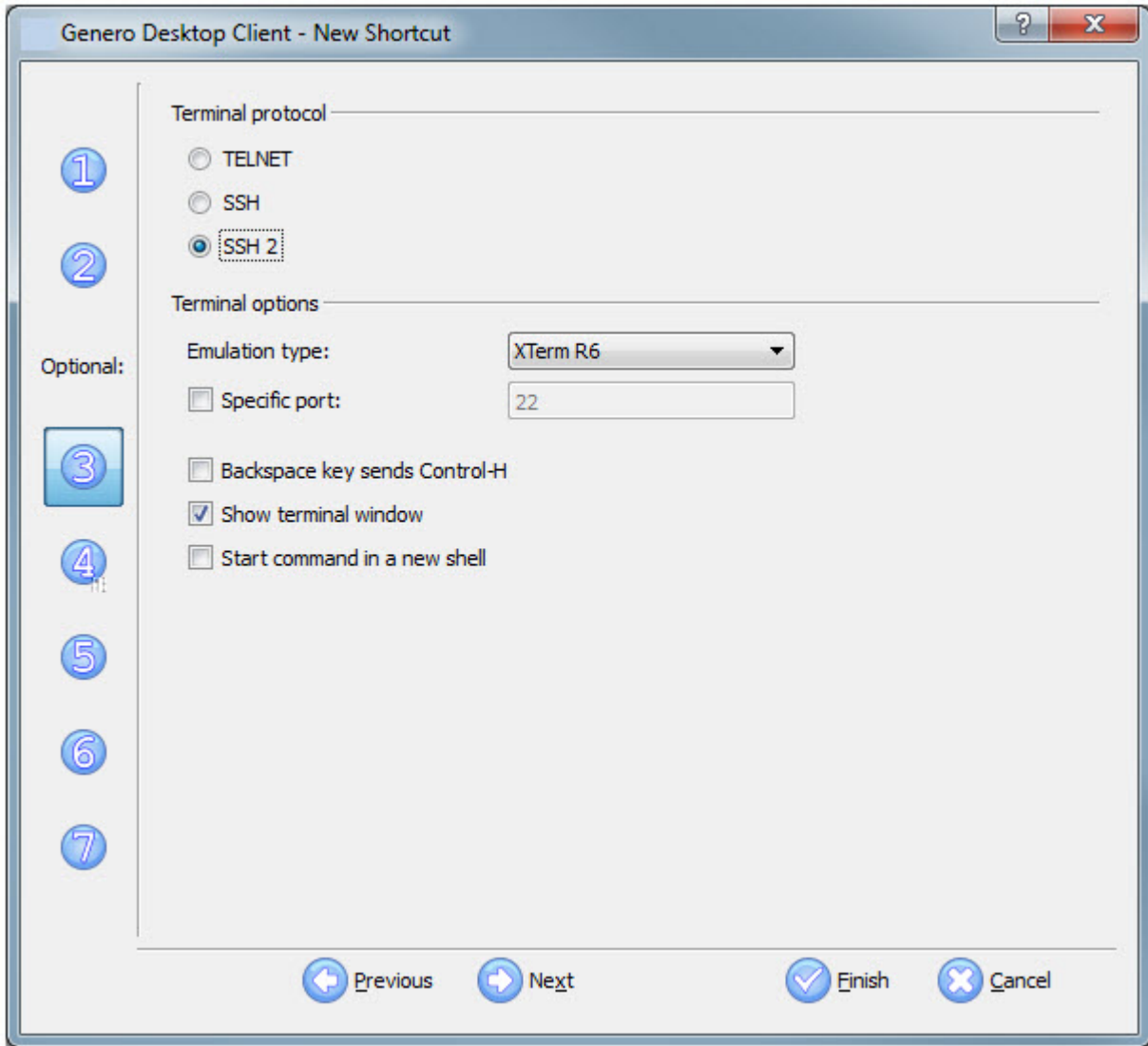


Figure 6-18. The Genero Desktop Client shortcut entry for using ssh.

In the Command field, we have specified AUTOPOINT. This corresponds to an option near the end in the login script.



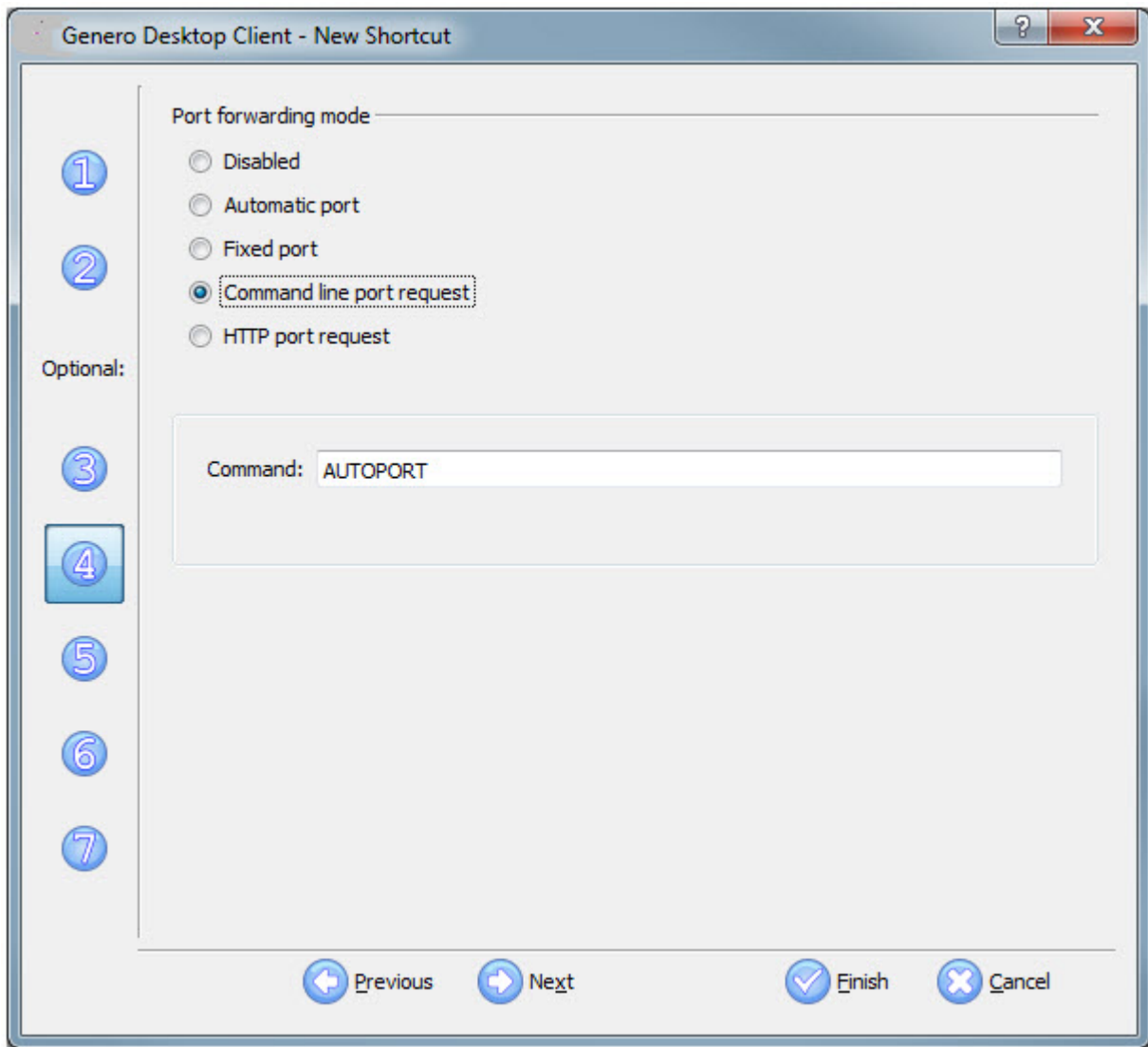


Figure 6-19. Setting AUTOPORT

When the login script receives "AUTOPORT", it executes a program called `autoportfind`. The `-e` option will make it output a string like "FJSPORTFORWARD=*nnnn*" where *nnnn* is the port number provided by the operating system. The string matching rule we use looks for FJSPORTFORWARD= and retains the number following the =. This session is then closed and a new session is started using that number as the port to forward. It should not matter where in the sequence this rule is added.

You will also need to make an addition in **Terminal Strings**.

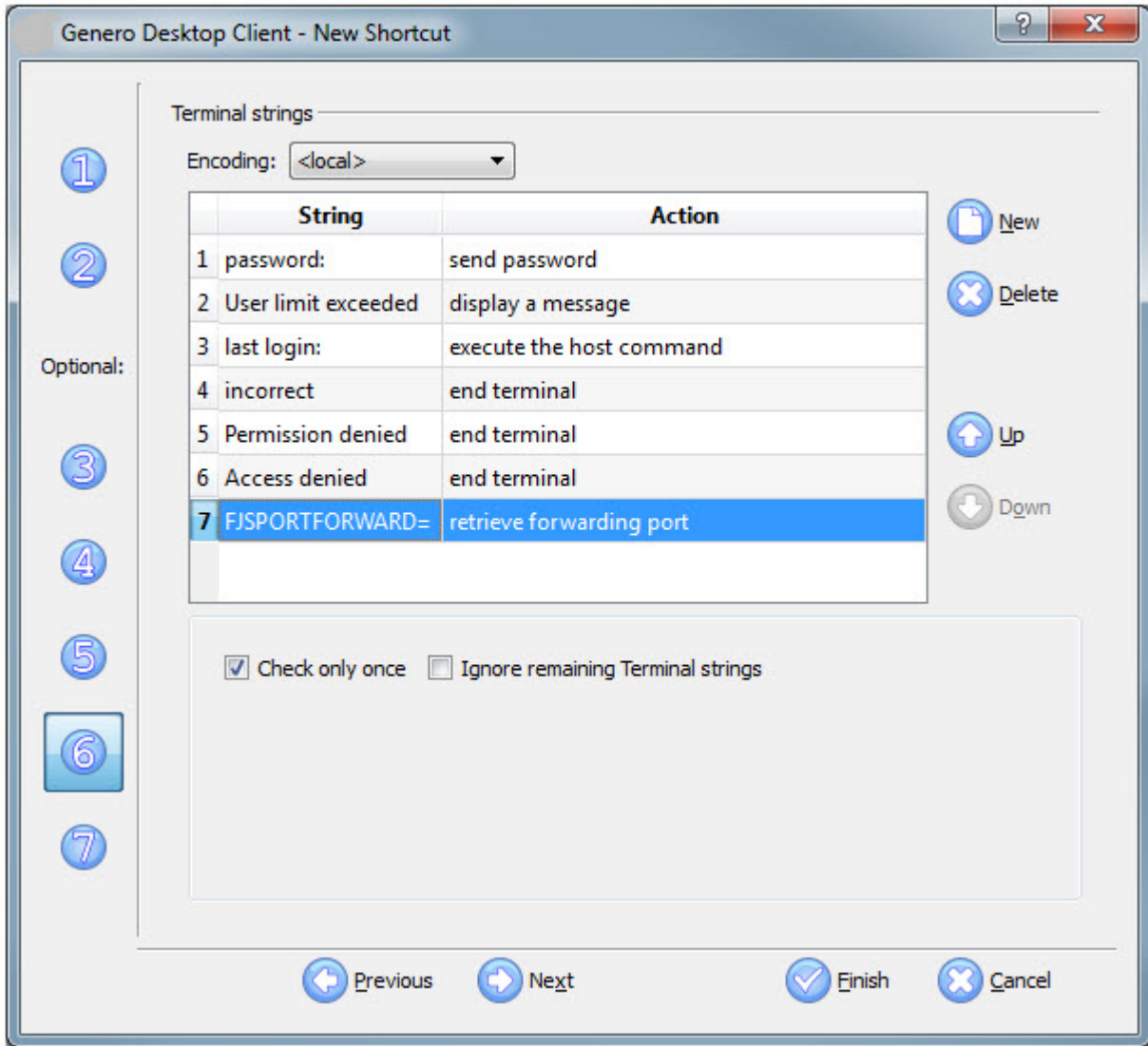


Figure 6-20. Configuring FJSPORTFORWARD in Terminal strings

Normally, the Command Line is passed to the shell that is started when a user logs in. Since we are using our shell script, the Command Line is where we specify the application to run, and pass the port number and the security fields. In our example we want to run the demo application. The command DEMO can be changed to your own application name, and an entry in the login script can then be added to start your application.

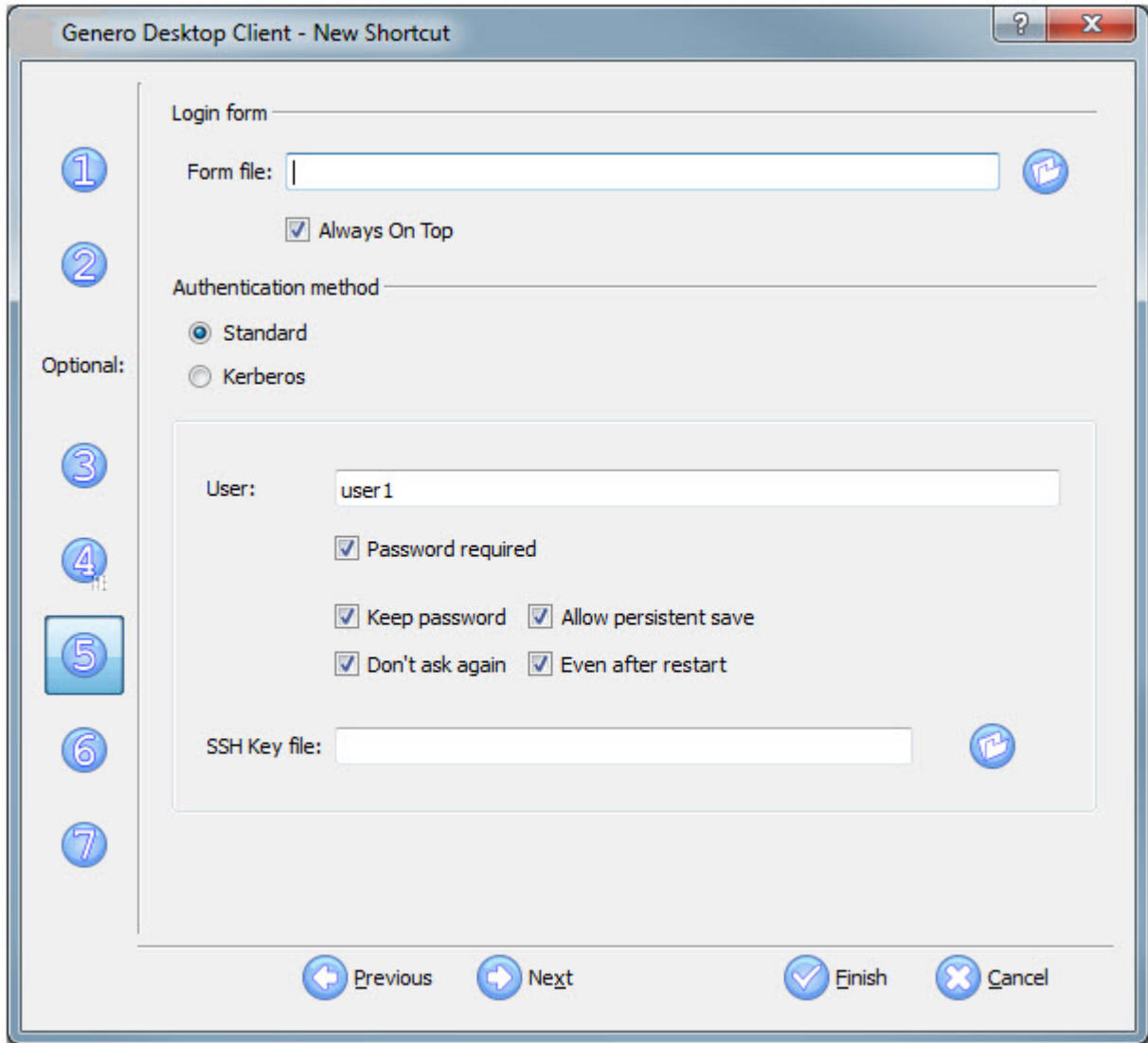


Figure 6-21. Run as user1

When the shortcut is run, it will log in using AUTOPOINT first. This will match a case statement in the script, and return a string "FJSPORTFORWARD=*nnnn*" where *nnnn* is a port number. Genero Desktop Client will then close the connection, and log in again using that port for the port to forward (tunnel) and pass it on the command line of the server @SRVNUM. This is what the login script uses to set the environment for the execution of the command DEMO. When using Port Forwarding, the server (127.0.0.1) is always the target for FGLSERVER (and therefore only the port number is needed).

## Setup telnet

Telnet don't offer port forwarding, so the setup is a bit simpler. But they also don't give the flexibility needed when going through firewalls, and offer no encryption or privacy like ssh.

You simply need to pass the required arguments via the command line, and the login script sets the environment and launches the application.

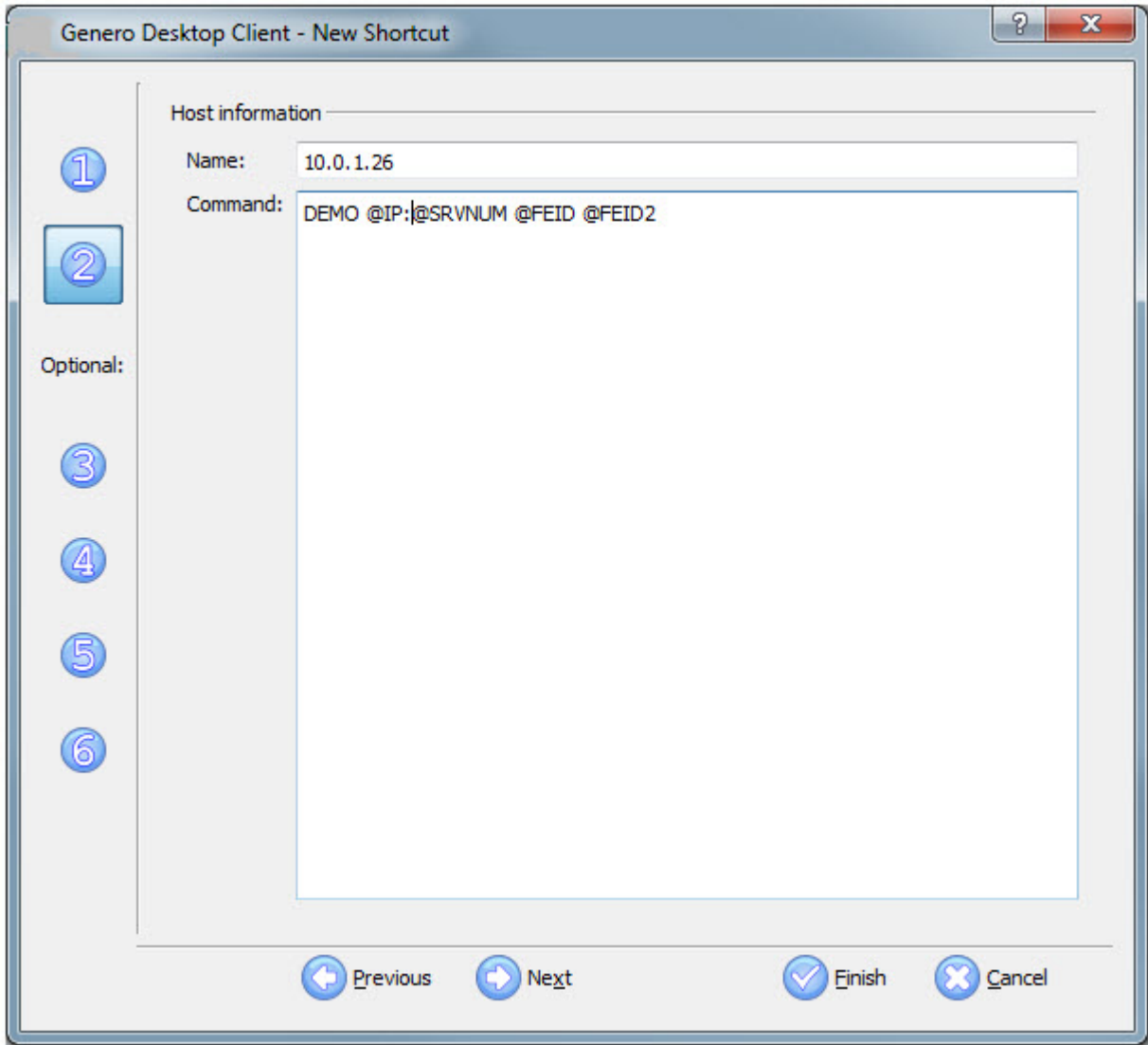


Figure 6-22. Specify the command line arguments for telnet

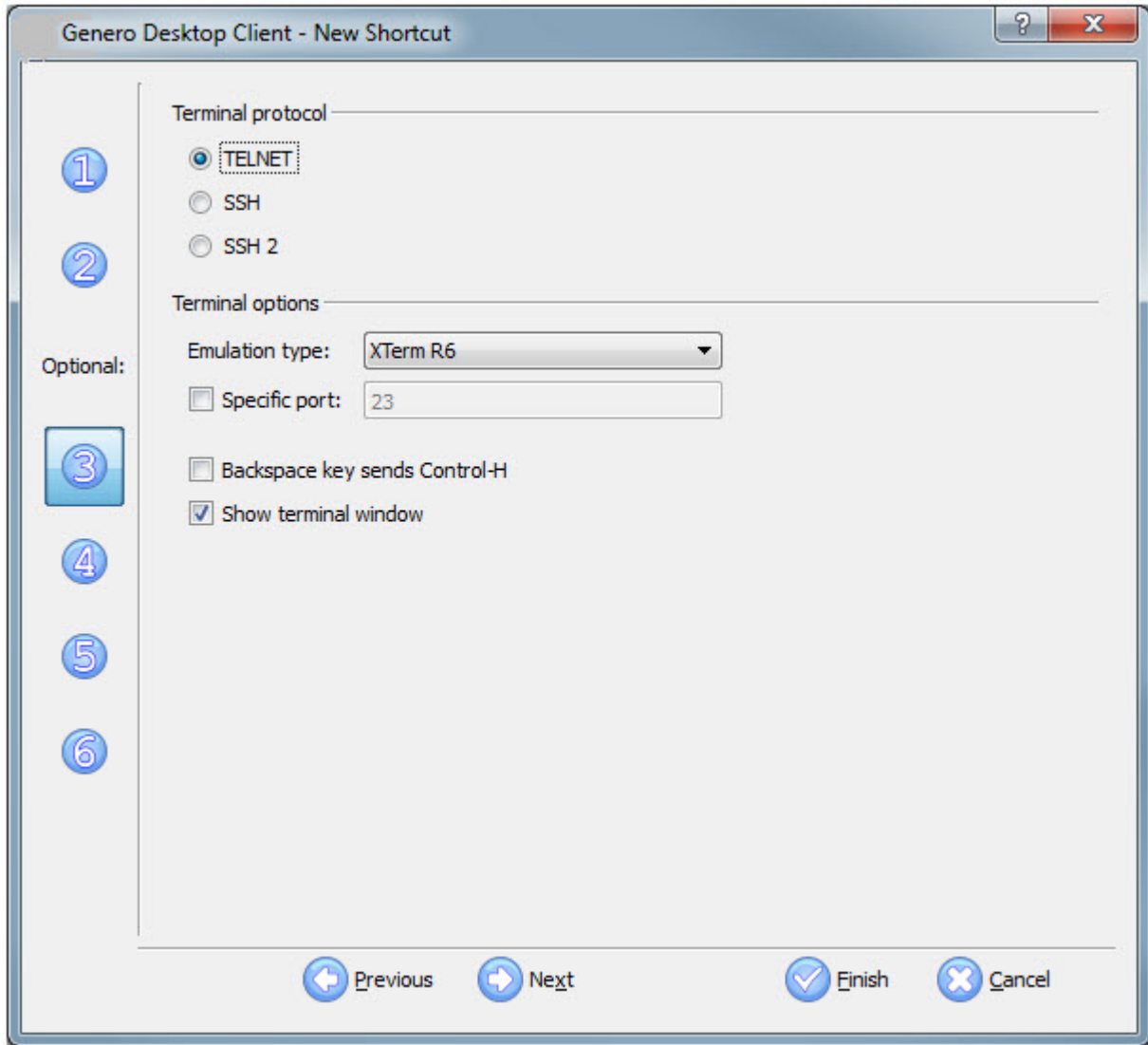


Figure 6-23. Select TELNET

With ssh and tunneling, the IP address is not needed because the tunnel is listening on the same server that will run the application. But with Telnet, we must pass the client machine's IP and port using @IP and @SRVNUM. The security values are passed as well, so the environment is complete. For the Genero Desktop Client to make use of the security values, you must start it with the option "-A 3" on the command line of the Genero Desktop Client. Put your application name in place of DEMO, and make an entry in the login script accordingly.

## Password management

These topics discuss secure server password management.

### Handling expired passwords

To handle expired passwords, edit the shortcut and add a filter under **Manage Connection Strings**. For the string **Your password has expired**, the action of **show terminal** should be set.

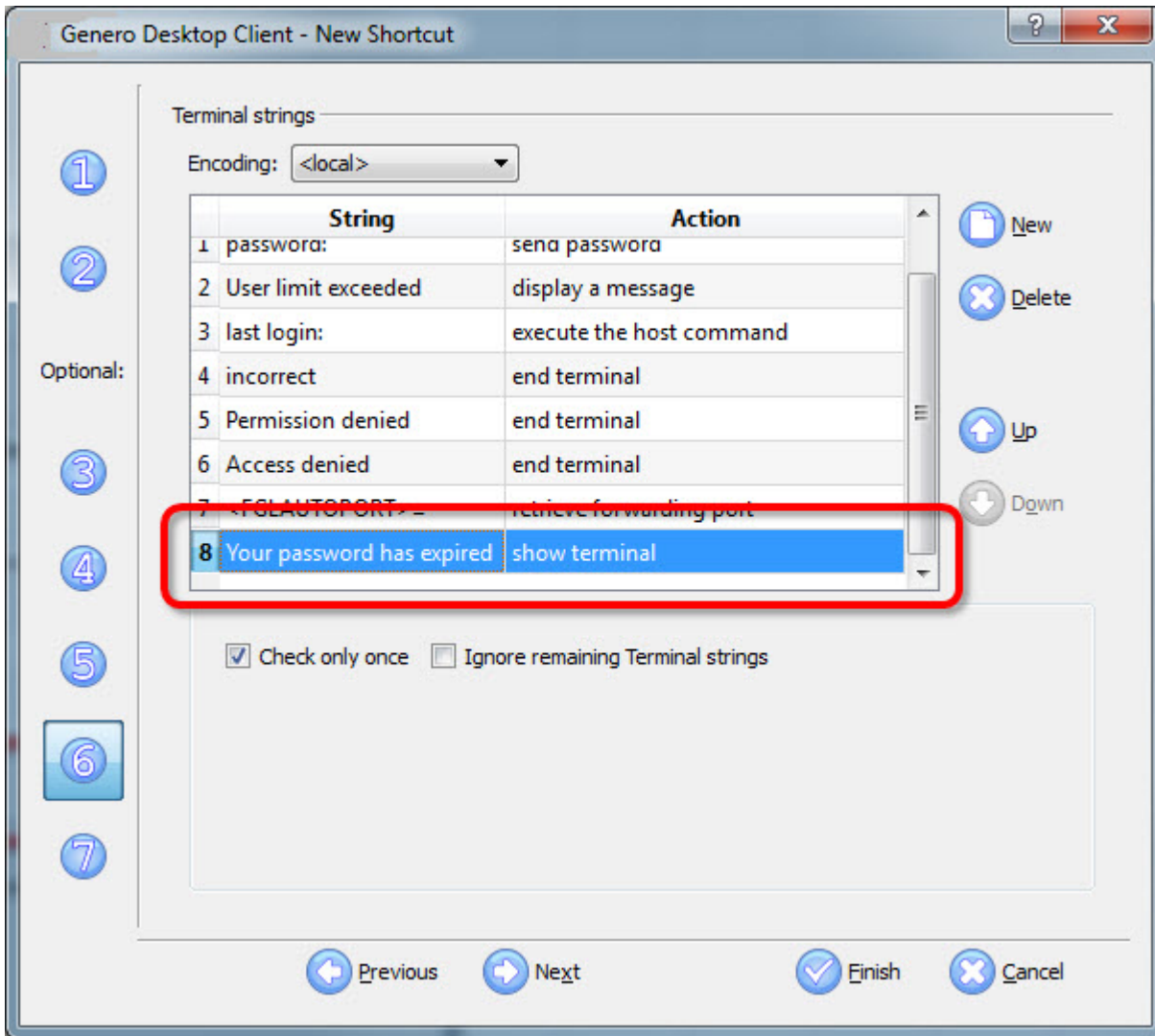


Figure 6-24. Setting Your password has expired

This rule looks for **Your password has expired** and open a text dialog window. Internally, the terminal window prompts for a new password from the server, as the existing password has expired. **Show the terminal** causes the Genero Desktop Client to display the server window, allowing the user to see the message and type in the correct passwords to complete the process. The window then closes and the user can click the shortcut once more and use the new password to start the application.

**Important:** The string entered in the Received String field must match the string displayed by the system. It is case-sensitive, where "Password has expired" does not match "password has expired". The string for an expired password may be different than the example shown above, based on your system. You should verify the string for an expired password that is returned by your system prior to implementing this solution.

### Changing passwords

Users may want to change their passwords prior to expiration. To allow for this functionality, provide a shortcut in the Genero Desktop Client that issues the password command. The sample login script uses a case statement that checks for

PASSWD. The specifics of the shortcut are as follows:

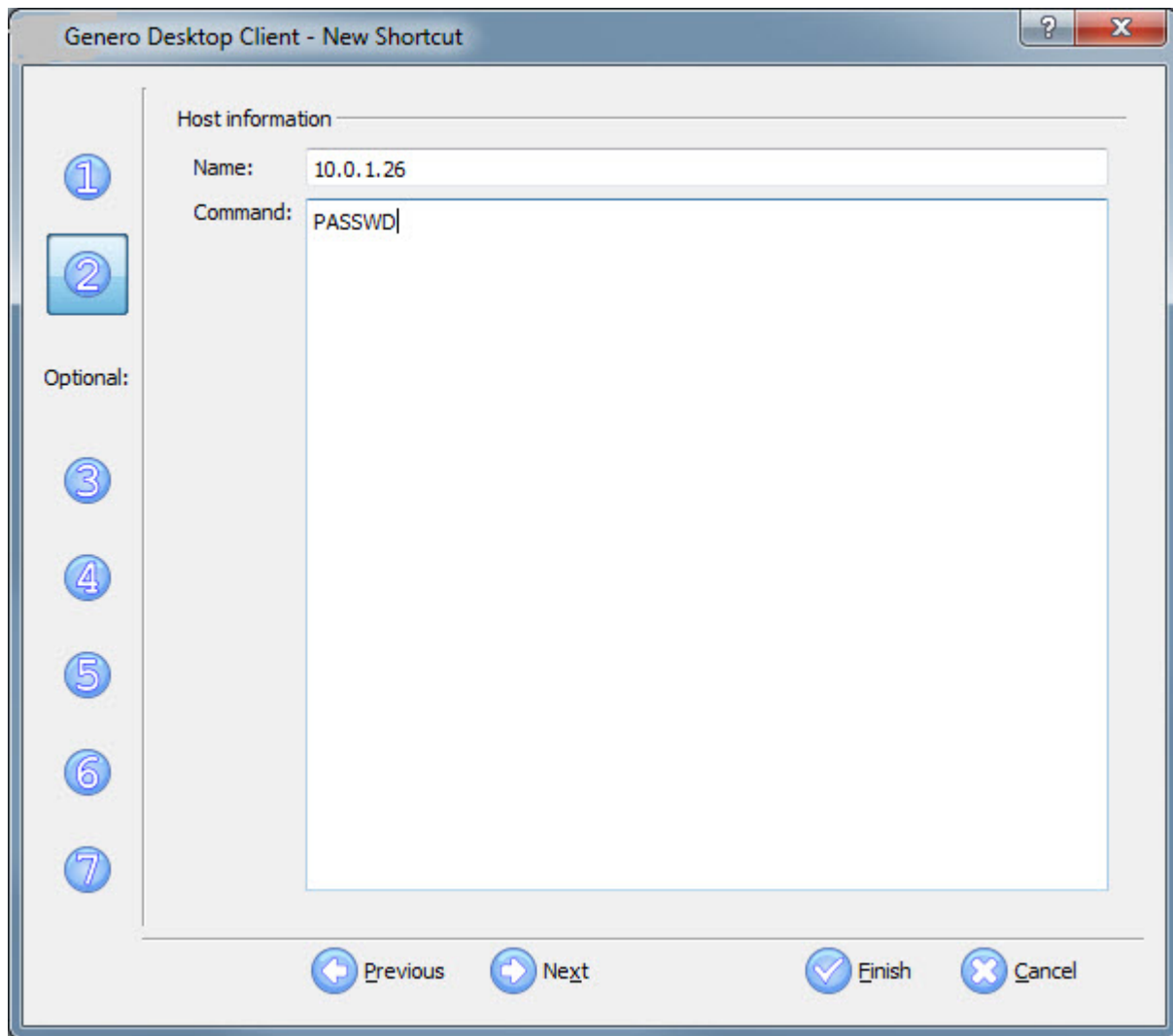


Figure 6-25. PASSWD command

## AUTOPORTFIND source code example

This is the source code used to produce the port number for tunnelling with ssh. It should compile with little or no modification and does not need to be run as root.

```
Autoportfind.c/*
```

```
Written by John A. Hobach, Dallas Texas, May 5th, 2004
```

```
The purpose of the application is to return a port number that  
will not be used for awhile. This port number can then be used  
by the Genero client for port forwarding.
```

```
The operating system assigns ports in a round  
robin fashion so the port assigned is unlikely to be used again  
very soon. This will give the GDC time to start ssh and use  
that port. The OS will automatically skip ports in use.
```

```
Revised 08/25/2004 Ver 2.1 to use bind() to get a port number  
assigned. It is assigned a port automatically from the  
operating system and we immediatly get it and return it.
```

```
Revised 10/25/2005 Ver 2.2 to support returning a port number  
within a given range. This is accomplished by requesting ports  
from the OS until it is within the range specified.
```

```
*/
```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#define USE_SOCKETS
#include "util.h"
char *progname;
static char *ver="autoportfind - Version 2.2, 2005-10-20";
static char *help=
"autoportfind [OPTION]\n"
"\n"
"Generate a port number for use with port forwarding.\n"
"\n"
"  -e, --env\n"
"      Send FJSPORTFORWARD=<port> to stdout.\n"
"\n"
"  -r      Cycle through port assignments to determine which ports\n"
"          the OS assigns to ports when originating connections.\n"
"  -u n    Upper limit. Request port numbers until one is returned\n"
"          below 'n'.\n"
"  -l n    Lower limit. Request port numbers until one is returned\n"
"          above 'n'.\n"
"  -h      Display this help message.\n"
"  -v      Display the version number.\n"
;
main(int argc, char **argv) {
    int sockfd, connected_socket, retval;
    int size, x, outofrange;
    int range_flag=0, env_flag=0;
    unsigned int    port, startport, highest,
                   lowest, cycle, direction,
                   llimit=0, ulimit=~0;
    int reuse_addr=1;
    char **arg;
    struct sockaddr in serv_addr;
    progname=argv[0];
    arg=argv;
    while (--argc) {
        ++arg;
        if (!strcmp(*arg, "-r") || !strcmp(*arg, "--range")) {
            range_flag=1;
        } else if (!strcmp(*arg, "-e") || !strcmp(*arg, "--env")) {
            env_flag=1;
        } else if (!strcmp(*arg, "-u")) {
            ++arg;
            if (argc == 1 || *arg[0] == '-') {
                fprintf(stderr, "%s: Value missing for -u\n", progname);
                exit(1);
            }
            --argc;
            ulimit=atol(*arg);
        } else if (!strcmp(*arg, "-l")) {
            ++arg;
            if (argc == 1 || *arg[0] == '-') {
                fprintf(stderr, "%s: Value missing for -l\n", progname);
                exit(1);
            }
            --argc;
            llimit=atol(*arg);
        } else if (!strcmp(*arg, "-v")) {
            printf("%s\n", ver);
            exit(0);
        } else if (!strcmp(*arg, "-h") || !strcmp(*arg, "--help")) {
            printf("%s", help);
            exit(0);
        }
    }
}

```



```

        } else {
            fprintf(stderr,"%s:Unknown argument '%s'\n",
                progname, *arg);
            exit(1);
        }
    }
    lowest=~0;
    highest=0;
    startport=0;
    cycle=0;
    direction=1;
    do {
        outofrange=0;
        memset((char*) &serv_addr,0,sizeof(serv_addr));
        serv_addr.sin_family=AF_INET;
        serv_addr.sin_port=0;          /* allow system to assign */
        serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
        if (sockfd < 0) {
            perror("socket");
            close(sockfd);
            exit(1);
        }
        if (bind(sockfd, (struct sockaddr *) &serv_addr,
            sizeof(serv_addr)) < 0) {
            perror("bind");
            close(sockfd);
            exit(1);
        }
        size=sizeof(serv_addr);
        if (getsockname(sockfd, (struct sockaddr *) &serv_addr,
            &size) == -1) {
            perror("getsockname");
            exit(errno);
        }
        if (range_flag) {
            port=ntohs(serv_addr.sin_port);
            if (!startport) startport=port;
            if (port > highest) highest=port;
            if (port < lowest) lowest=port;
            if (direction==0 && port <= startport) {
                cycle++;
                direction=1;
            } else if (direction==1 && port >= startport) {
                cycle++;
                direction=0;
            }
        } else {
            port=ntohs(serv_addr.sin_port);
            if (port > llimit && port < ulimit) {
                if (env_flag) printf("FJSPORTFORWARD=");
                printf("%d\n",ntohs(serv_addr.sin_port));
            } else
                outofrange=1;
        }
        close(sockfd);
    } while ((range_flag && cycle < 3) || outofrange);
    if (range_flag)
        printf("Lowest port: %lu\nHighest port: %lu\n",lowest,highest);
    exit(0);
}

---
Util.h
#ifdef UTIL_H
#define UTIL_H
#endif
MAX

```

```

# define MAX(a,b) a>b?a:b
#endif
#ifdef USE_SOCKETS
# ifdef _WIN32
#   include <winsock.h>
# else
#   include <sys/types.h>
#   include <sys/socket.h>
#   include <netinet/in.h> /* struct sockaddr_in, ... */
#   include <netinet/tcp.h> /* TCP_NODELAY, ... */
#   include <arpa/inet.h> /* inet_addr, inet_ntoa, inet_aton */
#   include <netdb.h> /* gethostbyname */
# endif
#endif

#ifdef _WIN32
# define SOCKLEN_T int
#endif

#ifdef __osf__
# define SOCKLEN_T int
#endif

#ifdef AIX
# ifdef USE_SOCKETS
#   include <sys/ioctl.h>
#   include <sys/time.h>
#   include <sys/select.h>
# endif
# define SOCKLEN_T socklen_t
#endif

#if defined (M_I386)
/* SCO */
# ifdef USE_SOCKETS
#   include <sys/ioctl.h>
#   include <sys/time.h>
#   include <sys/select.h>
# endif
# define SOCKLEN_T int
#endif

#ifdef linux
# define SOCKLEN_T socklen_t
#endif

#ifdef sun
# if defined USE_SOCKETS
#   undef USE_SYS_SOCKETIO
#   define USE_SYS_SOCKETIO
# endif
# define SOCKLEN_T int
#endif

#ifdef __hpux
# define SOCKLEN_T int
#endif

#ifndef SOCKLEN_T
# define SOCKLEN_T size_t
#endif

#ifndef MSG_DONTWAIT
# define MSG_DONTWAIT 0
#endif

#endif

```

## Login script

This is an example of the login script that is executed when users log in. It is intended to be an example, and we expect you to tailor it according to your needs. The login script is invoked via the `/etc/passwd` file.

```
#!/bin/sh

# Invoked directly by login mechanism such as telnetd, or sshd.
# This file is specified in the /etc/passwd file as being the shell. This
# gives us the control we need for users that should never be allowed a
# shell prompt.
#
# For backward compatibility we check to see if we are coming from a
# non-sshd source. If so then we invoke the shell as usual and have
# it source all the login scripts
#
# Arguments passed are <COMMAND> <PORT> <FEID> <FEID2>
#
# <COMMAND> string must match the case statements.
#

# set your env vars here
export FGLDIR=/fjs/f4gl/genero-training
export FGLRUN=fglrun
export FGLGUI=1

# The command line arguments passed from the GDC will be here. If there
# aren't any then we abort.

if [[ "$SSH_TTY" == "" && "$SSH_CONNECTION" == "" ]]
then
# coming in from telnet

echo -n "$ " # fake shell prompt for GDC
read APPLICATION FGLSERVER _FGLFEID _FGLFEID2

if [[ "$APPLICATION" == "" ]]
then
echo "exiting due to bad arguments"
sleep 5 # give time to view error because window will close
exit 0
fi

export FGLSERVER
export _FGLFEID
export _FGLFEID2

else
# coming in from ssh and sshd

if [[ "$1" == "" || "$1" != "-c" ]]
then
echo "exiting due to bad arguments"
sleep 5 # give time to view error because window will close
exit 0
fi

shift
args=(`echo $1`)
export APPLICATION="${args[0]}"
export FGLSERVER="127.0.0.1:${args[1]}"
export _FGLFEID="${args[2]}"
export _FGLFEID2="${args[3]}"
```

```

fi

#echo "APPLICATION=$APPLICATION"
#echo "FGLSERVER=$FGLSERVER"

# Add case statements according to 1st value passed from the GDC command line.
# Never execute the value passed directly as this would be a security hole
# allowing the client to dictate what gets run.
#
case "$APPLICATION" in
YOURAPP) cd $FGLDIR/demo
/bin/bash --login -c "$FGLRUN demo"
;;

DEMO) cd $FGLDIR/demo
$FGLDIR/bin/$FGLRUN demo
;;

# SHELL) /bin/bash # don't leave this in for production
# ;;

AUTOPORT) /home/portfind/autoportfind -e
exit 0
;;

PASSWD) /usr/bin/passwd
exit 0
;;

*) echo "Unknown application '$APPLICATION'"
sleep 5 # allow time to read message
;;
esac

```

---

## SSH Configuration Troubleshooting

These topics discuss possible configuration issues when implementing SSH.

### Duplicate port forward number

If you have more than one client using the same Port Forward port number, the application display could go to the wrong client. This is an expected result, so precautions need to be taken to prevent this. Make sure that each client using port forwarding to a particular server uses a different port forward port number.

## No Duplicate Port Forward Numbers

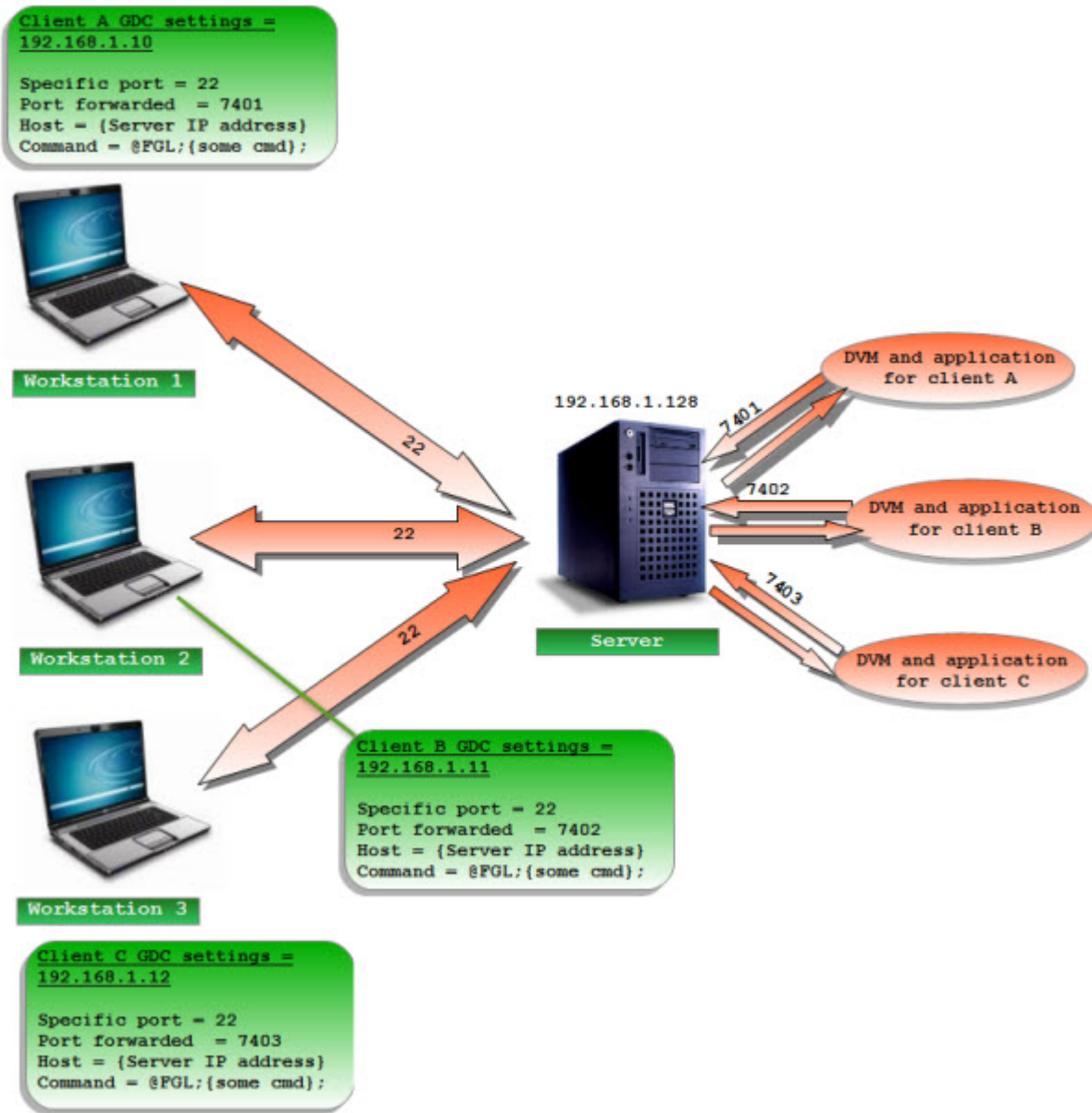


Figure 6-26. No Duplicate Port Forward Numbers

The `sshd` sets up a tunnel and listens on the port that is specified in the port forward field. This allows the Genero application (or DVM) to connect to that port and have the GUI data sent to the client. Only one listener can be listening on any port at a time. Each client needs to use a unique port number to avoid any problems.

**Note:** An automatic configuration solution is being worked on but has not been released at the time of this writing.

## Wireless systems

The latest technology to use is 802.11(a,b or g). This is great at avoiding the wire mess, but there is a new risk. Under Windows, if you are using a plugged in or built-in wireless card, the interface goes offline if the signal is lost for even a

second. When this happens, it is treated similar to unplugging your network cable. The Windows drivers report to the network stack that the interface is now offline, and everything associated with that interface is removed. If an application has an open channel, it is signaled that it has closed. As a result, you lose all your connections and must wait for your signal to return in order to log in again.

A possible workaround is to use an external wireless device that doesn't take the connection down when the signal is lost. This works because it doesn't look like the cable was unplugged when it loses signal, so Windows doesn't know there is a problem. When the signal returns, everything works just as before.

## Need to change the port that GDC listens on

Why would you want to change the port that GDC listens on?

You may need to run several versions of the GDC on the same machine. Since each one must have its own listening port, Genero allows you to specify the port. If you run more than one and don't specify the port, Genero opens the next available port. For example, the first instance would open 6400, the next instance would open 6401.

```
>gdc -< The port assigned would be 6400
>gdc -n -< The port assigned would be 6401
>gdc -n -p 7400 -< The port assigned would be 7400
>gdc -n -p 7400 -< The port assigned would be 7401
>gdc -q -p 7400 -< GDC won't start since the port 7400 is already
assigned
```

Another reason to change ports might be that you can't use the ssh functionality. What if you haven't installed the SSH package yet, but you have more than one client behind the same firewall router? You can add rules to the router to send 6400 to the first client, 6410 to the second client, and so on. Each client would be started with the corresponding `-p <port>`, and the router would make sure each client gets the connections intended for it.

## Sessions expiring

If you have sessions expire or applications that disappear, check for routers that expire sessions. Most likely, there is a firewall router in the path. If you are using a firewall router, check for session expiration timers for the ports used to get through the firewall. The expiration duration (KeepAlive) should be set greater than the interval set in your operating system. This is set to 2 hours as a default on most computers. The operating systems can be tuned to have shorter values, but it is usually easier to adjust the router; use a value of 2 hours and 10 minutes.

---

## GDC and Windows XP Service Pack 2

These topics identify issues to be aware of when running the Genero Desktop Client on Windows XP Service Pack 2.

### GDC and Windows XP Service Pack 2 Firewall Configuration

Microsoft has added several security systems in Windows XP Service Pack 2 (SP2). The firewall included in Windows XP has been improved and is now enabled by default. From the network point of view, GDC is a server: it listens on a defined port (6400 by default) for Runtime System connections.

When GDC starts, the firewall detects that it listens on port 6400 and warns the user: Press **Unblock** to allow the GDC to run correctly.

**Important:** Pressing **Keep Blocking** or **Ask Me Later** will keep GDC from working. Connections from the Runtime System will be blocked by the firewall.

If **Keep Blocking** has been pressed by mistake, this parameter can be changed in the Firewall settings (Control Panel).

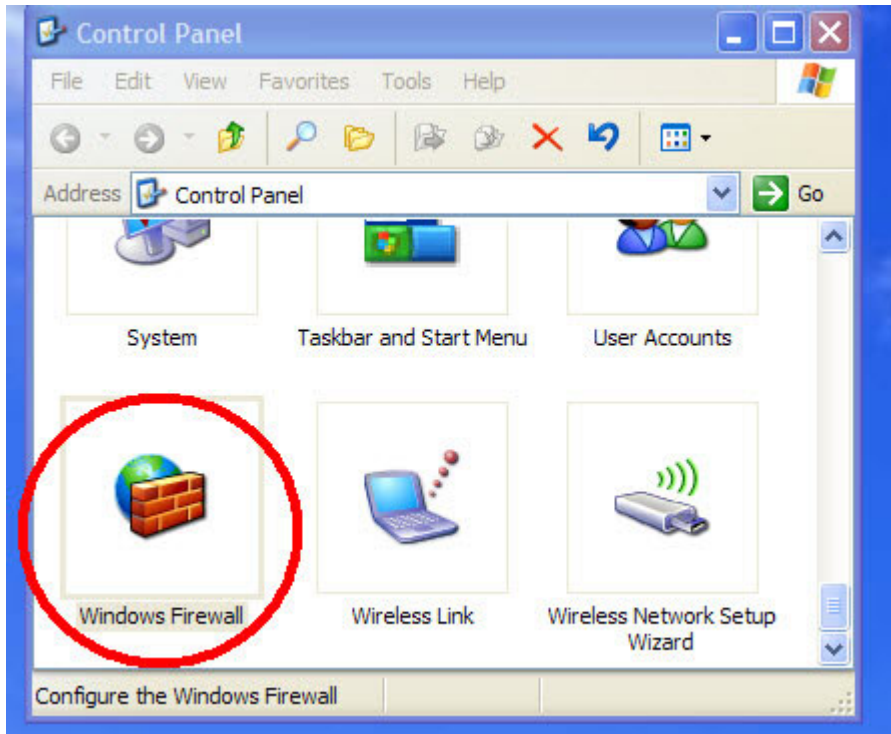


Figure 6-27. Control Panel; Windows Firewall

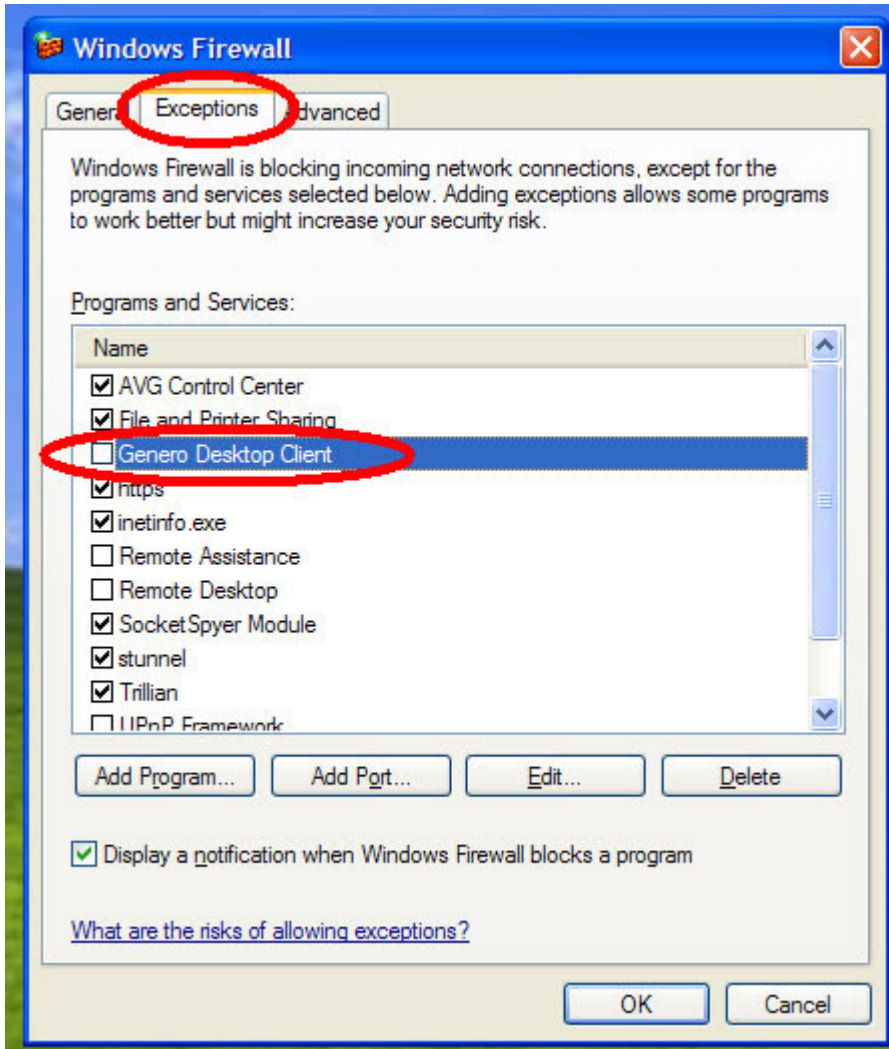


Figure 6-28. Windows Firewall; Exceptions Tab

In the **Exceptions** tab, Genero Desktop Client must be present and checked.

## GDC ActiveX and Windows XP Service Pack 2

Depending on your configuration, the installation of the ActiveX may be blocked by Windows Security.

To install the ActiveX, you have to click on the Information Bar and choose **Install ActiveX Control**.

The page will be reloaded and then you will be prompted to install GDC ActiveX.

Press **Install**. ActiveX will be installed and started. As ActiveX is also listening on port 6400 in the same way as the non-ActiveX version, the warning described in the firewall configuration will display.

---

## GDC and Windows Vista

These topics identify issues to be aware of when running the Genero Desktop Client on Windows Vista.



## User Account Control

One of the new features of Window Vista is the User Account Control (UAC). UAC prevents any software from silently hurting your system by prompting the user before any administrative actions such as:

- Installing a new program
- Modification of the registry

It requires a user with Standard User rights (users not in the *Administrator* group) to provide an Administrator login and password when running a program that performs system-level tasks. Administrator Users will only have to confirm their actions. More details can be found on the Microsoft Web site.

The User Account Control feature affects the Genero Desktop Client installation, as well as how GDC is run.

## Genero Desktop Client installation on Windows Vista

When the installation program starts, you'll be prompted to validate the installation. If you are not logged in as an administrator, you will be asked for an administrator password.

The installation then continues as in Windows XP.

## Running the Genero Desktop Client on Windows Vista

Once GDC is installed, the Windows Firewall will prompt the user to unblock the program, as described in GDC and Windows XP Service Pack 2.

Although most of the features of Genero Desktop Client will work out of the box on Windows Vista, some features will only work if you start GDC as administrator.

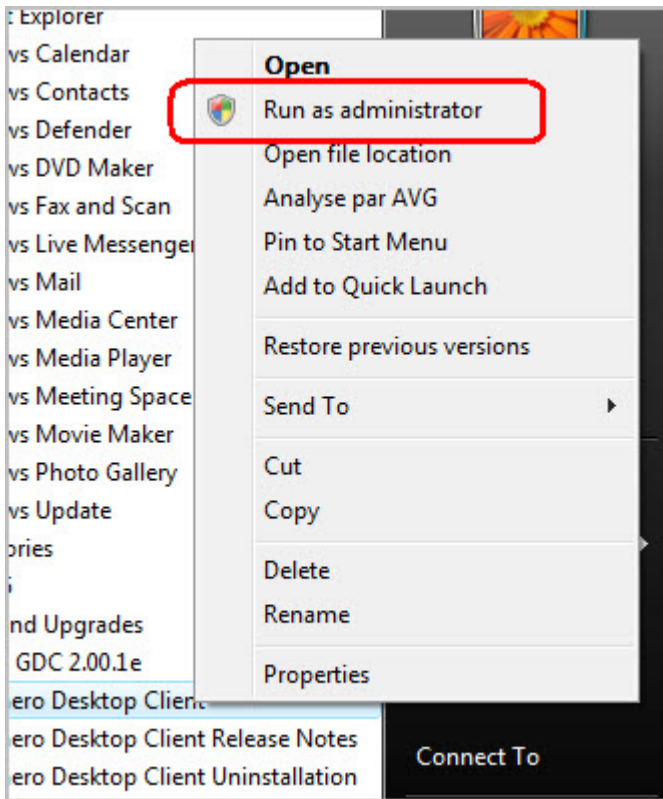


Figure 6-29. Run as administrator

Even an Administrator User has to run the program "as administrator". However, Administrator users can create a shortcut and specify in the **Compatibility** tab that this program is always run as an administrator.

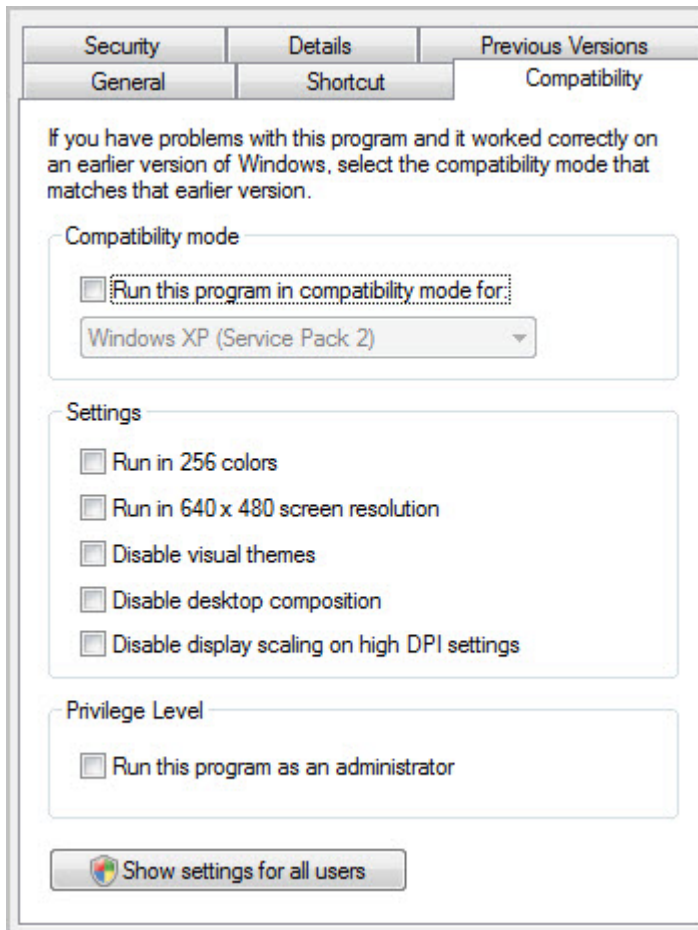


Figure 6-30. Compatibility tab

**Note:** Using the `-a` GDC command line option to run GDC in "admin mode", a mode where you can manage GDC shortcuts, is not the same as asking Windows Vista to start GDC "as administrator", a special mode where the program can perform system-level tasks.

## Genero Desktop Client features affected by the User Account Control

These topics describe GDC features affected by the UAC.

### Configuration

The folder `%Program Files%` is now protected with Windows Vista. So when starting GDC in normal mode, GDC will react as if the config file is read-only, and will only allow changes via the registry (local shortcuts). This may change in future versions of GDC.

### File Transfer

#### The ProgramFiles folder

As the folder `%Program Files%` is protected, Vista has introduced the concept of Virtual Store: instead of using the `%Program Files%` folder, programs will access a special directory. This means that if you use `FGL_PUTFILE` without a destination, the file will use the `%VirtualStore%` folder to transfer the file.

The following example would use the virtual folder %VirtualStore%. to transfer the file myFile.txt:

```
CALL FGL_PUTFILE("myFile.txt","myFile.txt")
```

## Uploading files

Vista will prevent the upload of some files, such as executables or DLLs.

Example:

```
MAIN
  DISPLAY "upload text file..."
  CALL FGL_PUTFILE("myFile.txt","myFile.txt")
  DISPLAY "... done"
  DISPLAY "upload DLL..."
  CALL FGL_PUTFILE("myFile.dll","myFile.dll")
  DISPLAY "... done"
END MAIN

$fglrun ft
upload text file...
... done
upload DLL...
Program stopped at 'ft.4gl', line number 7.
FORMS statement error number -8066.
Could not write destination file for file transfer.
```

Running GDC as administrator will allow GDC to upload such files to the system.

## Windows Vista and Genero Desktop Client ActiveX

These topics identify issues to be aware of when running the Genero Desktop Client ActiveX on Windows Vista.

### Genero Desktop Client ActiveX installation on Windows Vista

A lot of protection has been added to Vista to prevent spyware or other malicious programs from being installed without the agreement of the end user. Therefore, installing an ActiveX needs strong user agreement by default.

ActiveX is run by *Internet Add-on Installer*, a special tool used by Internet Explorer 7 to install ActiveX. Running this tool requires the agreement of the end-user and UAC Administrator User rights. When the Security Warning displays, click Install to continue.

The first time you start GDC ActiveX, two steps are performed:

- Deploying files (simple copy) in %Program Files%
- Registering the ActiveX (which is done by `gdc.exe /regserver`), to tell Internet Explorer how to execute the ActiveX.

The first step is done by *Internet Explorer Add-on Installer*, but the second requires IE to be started *as Administrator*:

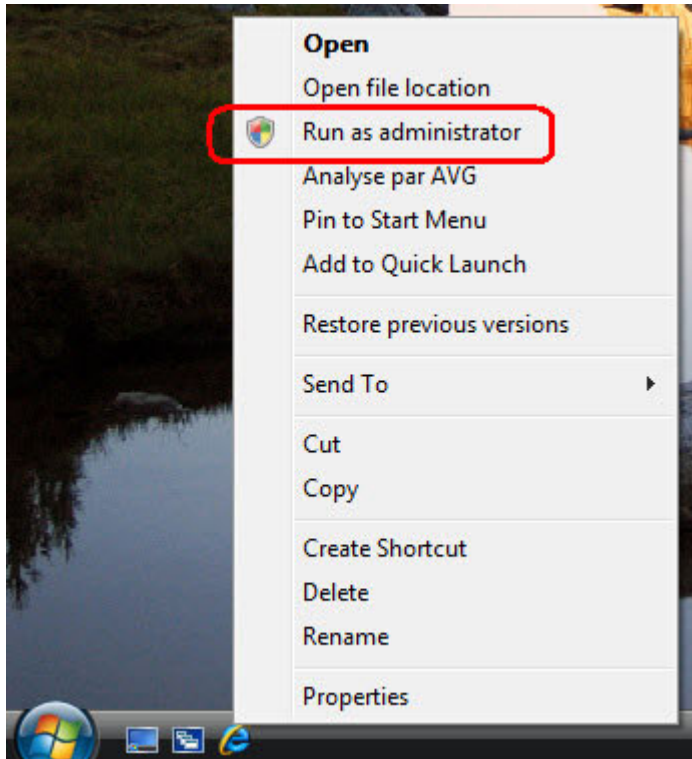


Figure 6-31. Run as administrator

If not, GDC will not be registered and Internet Explorer will not be able to load GDC ActiveX. The following message will display:

Installation of Genero Desktop Client ActiveX failed, object not registered. Note for Microsoft Windows Vista Users: Internet Explorer must be run as Administrator to be able to install ActiveX. Please start IE with right-click and *run as Administrator* and try again.

**Note:** This message has been introduced in the template in GDC 2.00.1e or greater.

### **Running the Genero Desktop Client ActiveX on Windows Vista**

It is not necessary to run IE as Administrator in order to use GDC ActiveX, only to install it. Once the ActiveX has been installed, the user will be prompted to allow Internet Explorer to start Genero Desktop Client ActiveX.

Depending on your Security Settings, Internet Explorer may not be able to start ActiveX ; the default security settings have been changed to allow less freedom to ActiveX for Vista. You may have to change these settings to allow Genero Desktop Client ActiveX to be used on your computer.

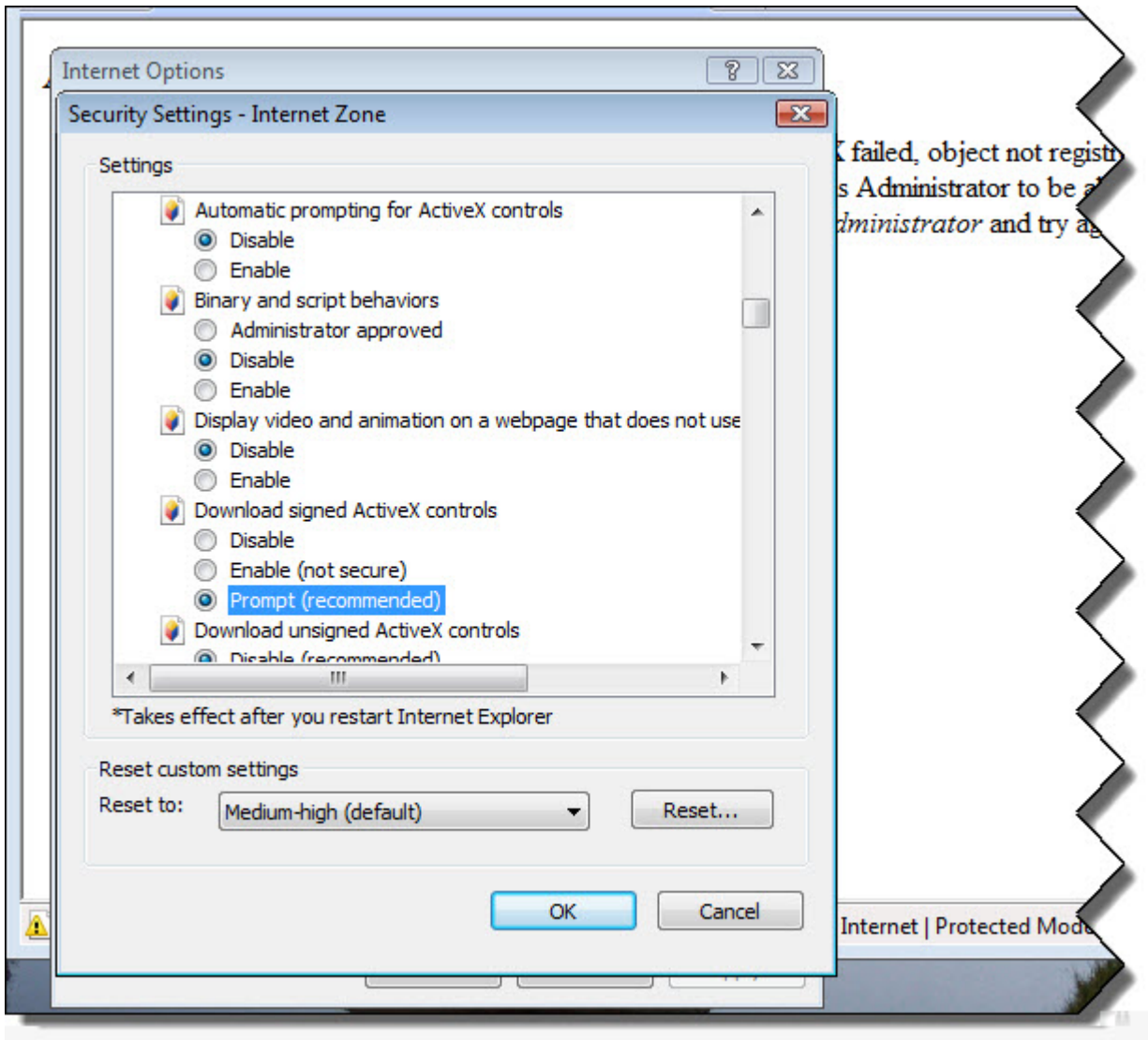


Figure 6-32. Security Settings, Download ActiveX controls

**Note:** The gdc.exe file has been signed in version 2.00.1e or greater.

---

## Chapter 7. Front End Extensions

These topics tell you all you need to know for creating and using front end extensions.

- “Using Front End Extensions”
- “Windows DDE Support” on page 7-5
- “Windows COM Support” on page 7-10
- “Windows Mail extension” on page 7-17

---

### Using Front End Extensions

The Front End allows you to call external functions. These functions are dynamically loaded by the front end when needed; they are within a DLL (Dynamic Linked Library under Windows systems), so (Shared Object under Linux), or dyLib (Dynamic Library under Mac Os X).

This allows you to create your own extensions and use them from your 4GL code. The parameters are sent to the front end, and a stack is used to transmit in and out parameters to the extension.

### How Front End Extensions work

This small tutorial on how to create your own extension explains the mechanism of Front End Extensions.

#### STEP 1: The 4GL code

Front end extensions can be called using the `ui.Interface.frontCall()` built-in function. See the Runtime System documentation for more information about this function.

```
CALL ui.Interface.frontCall(  
    module, function, <in-list>, <out-list>)
```

For example, if the extension is called `myExt`, the function used is `makeSum`, and the function takes 2 parameters in and returns an integer and a string:

```
DEFINE a,b INTEGER -- the two IN parameters  
DEFINE c INTEGER -- the integer returned  
DEFINE res STRING -- the string returned
```

```
CALL ui.Interface.frontCall( "myExt", "makeSum", [a,b], [c,res])
```

#### STEP 2: Internal GDC mechanism - the stack

To transmit parameters to the extension, GDC uses a stack; `a` and `b` will then be pushed on that stack.

#### STEP 3: Call the external function

All the information needed by the front end is transmitted to the extension using a front end interface structure. This structure contains a list of function pointers to:

- manage the stack (push or pop for each handled data type)
- get information on the function (number of IN or OUT parameters)

- get information about the front end

For this reason the prototype of each function should be the same.

#### **STEP 4: The extension function is running**

The classic process is:

1. Check that the number of parameters is correct
2. Retrieve the parameters using the stack functions
3. Do what the function has to do
4. Use the stack functions to stack the return values
5. Return 0 in case of success

#### **STEP 5: Internal GDC mechanism - the stack**

The GDC uses the stack to un-stack the returned values and transmit them to the Runtime System, which dispatches the values into the right variable.

### **The Initialize and Finalize functions**

The Front End automatically performs two functions when using a Dynamic Library:

- `void initialize();`  
called the first time the library has been loaded.
- `void finalize();`  
called when the front end stops.

These two functions allow you to have better control of your extension.

### **Front End Extension function prototype**

A Front End Extension Function must have the following prototype:

```
int <name> ( const struct frontEndInterface &<fx>);
```

**Note:**

1. <name> is the name of your function.
2. <fx> is the structure for the frontEndInterface
3. This function return 0 if it is successful, -1 if not.

Parameters are transmitted via a stack. This stack can be managed using the functions provided by the frontEndInterface structure. `getParamCount();` and `getReturnCount();` can be used to check if the number of parameters is correct.

### **Front End Interface structure**

The Front End will dynamically load the library and call the given function. IN and OUT parameters are managed using a stack. All the operations concerning this stack are managed by functions within the Front End. Pointers to these functions are gathered into a front end interface structure:

```
struct frontEndInterface
{
    short (* getParamCount) ();
    short (* getReturnCount) ();
    void ( * popInteger) (long & , short & );
    void ( * pushInteger) (const long , short );
    void ( * popString) (char *, short &, short &);
```



```

void ( * pushString) (const char *,short , short );
void ( * getFrontEndEnv) (const char * , char *, short & );
/*new in 2.00*/
void ( * popWString) (wchar_t *, short &, short &);
void ( * pushWString) (const wchar_t*,short , short );
};

```

Table 7-1. Front end interface functions

Function	Description
short getParamCount();	This function returns the number of parameters given to the function called.
short getReturnCount();	This function returns the number of returning values of the function called.
void popInteger( long & value, short & isNull );	This function is used to get an integer from the stack: value is the reference to where the popped integer will be set, isNull indicates whether the parameter is null.
void pushInteger( const long value, short isNull );	This function is used to push an integer on the stack: value is the value of the integer, isNull indicates whether the value is null.
void popString( char * value, short & length, short & isNull );	This function is used to get a string from the stack: value is the pointer where the popped string will be set, length is the length of the string, isNull indicates whether the parameter is null.
void pushString( const char * value, short length, short isNull );	This function is used to push a string on the stack: value is the value of the string,length the length of the string, isNull indicates whether the parameter is null. A length of -1 indicates that the length is detected based on the content of the string.
void ( * getFrontEndEnv )(const char * , char *, short & );	This function is used to get information from the front end. The table in Environment Variables indicates which front end environment variable is set.
void ( * popWString) (wchar_t *value, short &length, short &isNull);	This function is used to get a unicode string from the stack: value is the pointer where the popped string will be set, length is the length of the string, isNull indicates whether the parameter is null.
void ( * pushWString) (wchar_t *value, short length, short isNull);	This function is used to push a unicode string on the stack: value is the value of the string, length the length of the string, isNull indicates whether the parameter is null. A length of -1 indicates that the length is detected based on the content of the string.

## Front End environment variables

Table 7-2. Front end environment variables

Name	Meaning
frontEndPath	The path where the front end is installed

## Front End example: create a basic extension

This very basic extension illustrates how to create a DLL. It has been created using Visual C++ 6.

This DLL has only one function called makeSum. Two Integers are given in parameters, and the function returns a String and an Integer. The Integer contains the sum, the String a small text.

### Header file : myDLL.h

```
/*the interface structure*/struct frontEndInterface {
short (* getParamCount) ();
short (* getReturnCount) ();
void (* popInteger) (long & , short & );
void (* pushInteger) (const long , short );
void (* popString) (char *, short &, short &);
void (* pushString) (const char *,short , short );
void (* getFrontEndEnv(const char *, char *, short&);
void (* popWString) (wchar_t *, short &, short &);
void (* pushWString) (const wchar_t*,short , short );
};
// a small macro used to declare the "exportable" functions#ifdef WIN32
//dlexport is only valid (and mandatory) under windows*/
#define EXPORT extern "C" __declspec(dllexport)
#else
#define EXPORT extern "C"
#endif

// the functions we want to be available from the front endEXPORT void initialize();
EXPORT void finalize();
EXPORT int makeSum(const frontEndInterface &fx);
```

### Source file : myDLL.cpp// some includes

```
#include "myDLL.h"
#include <stdio.h>

// this function will be called by the Front End the first time the DLL is
loaded
void initialize() {
}

// this function will be called by the Front End when the Front End stops
void finalize() {
}

// our makeSum function. // --> in parameters the front End Interface
Structure
int makeSum(const struct frontEndInterface &fx) {

// initialize the status
short status = -1;

// check if the in and out parameters are correct
if (fx.getParamCount() == 2 && fx.getReturnCount() == 2) {
long param1, param2;
short isNull1, isNull2;

// get from the stack each parameter
```

```

    fx.popInteger(param2, isNull2);
    fx.popInteger(param1, isNull1);

    // check if they are not null
    if (!isNull1 && !isNull2) {

    // create the answer
        long sum = param1 + param2;
        char msg[255];
        sprintf(msg, "%d + %d = %d", param1, param2, sum);

    // push the answer on the stack
        fx.pushInteger(sum, 0);
        fx.pushString(msg, strlen(msg), 0);

    // successful -> status = 0
        status = 0;
    }
    }
    return status;
}

```

Running the program gives the following output:

```

$fglrun testDLL
1 + 3 = 4
4

```

---

## Windows DDE Support

Dynamic Data Exchange (DDE) is a form of inter-process communication implemented by Microsoft for Windows platforms. DDE uses shared memory to exchange data between applications. Applications can use DDE for one-time data transfers and for ongoing exchanges in applications that send updates to one another as new data becomes available.

Please refer to your Microsoft documentation for DDE compatibility between existing versions. As an example, DDE commands were changed between Office 97 and Office 98.

We provide a DDE interface as a Front-End Extension: WinDDE.DLL

### Using the DDE API

With DDE Support, you can invoke a Windows application and send or receive data to or from it. To use this functionality, the program must use the Windows Front End.

Before using the DDE functions, the TCP communication channel between the application and the front end must be established with a display (OPEN WINDOW, MENU, DISPLAY TO).

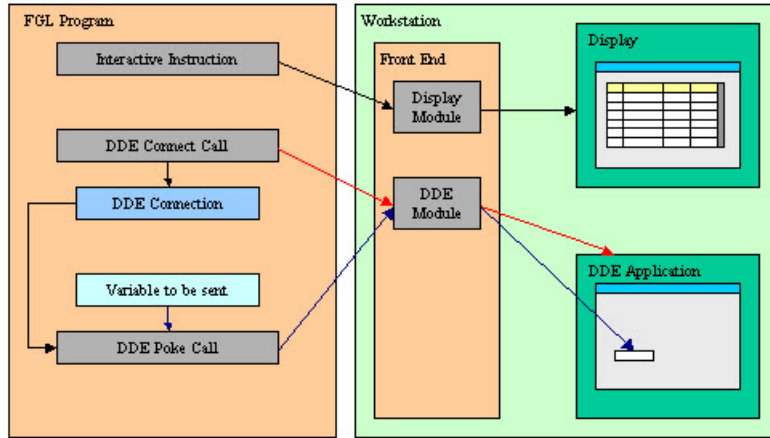


Figure 7-1. The four-part procedure of the DDE API

The DDE API is used in a four-part procedure, as described in the following steps:

1. The application sends to the Front End the DDE order using the TCP/IP channel.
2. The Front End executes the DDE order and sends the data to the Windows application through the DDE API.
3. The Windows application executes the command and sends the result, which can be data or an error code, to the Front End.
4. The Windows Front End sends back the result to the application using the TCP/IP channel.

A DDE connection is uniquely identified by two values: The name of the DDE Application and the document. Most DDE functions require these two values to identify the DDE source or target.

## The DDE API function list

The DDE API is based on the front call technique as described in Front End Functions. All DDE functions are grouped in the **WINDDE** front end function module.

Table 7-3. DDE API functions

Function name	Description
DDEConnect	This function opens a DDE connection.
DDEExecute	This function executes a command in the specified program.
DDEFinish	This function closes a DDE connection.
DDEFinishAll	This function closes all DDE connections, as well as the DDE server program.
DDEError	This function returns DDE error information about the last DDE operation.
DDEPeek	This function retrieves data from the specified program and document using the DDE channel.
DDEPoke	This function sends data to the specified program and document using the DDE channel.

## DDEConnect

### Purpose:

This function opens a DDE connection.

### Syntax:

```
CALL ui.Interface.frontCall("WINDDE","DDEConnect",  
    [ program, document, encoding ], [ result ] )
```

### Note:

- *program* is the name of the DDE application.
- *document* is the document that is to be opened.
- *encoding* is an optional parameter. It allows to force the encoding to use between ASCII and wide char/unicode. When not specified, WinDDE will try to retrieve the correct encoding by itself. Possible values are:
  - UNICODE
  - ASCII
- *result* is an integer variable receiving the status.
- *result* is TRUE if the function succeeded, FALSE otherwise.
- If the function failed, use DDEError to get the description of the error.

### Warnings:

- If the function failed with DMLERR\_NO\_CONV\_ESTABLISHED, then the DDE application was probably not running. Use the execute or shellexec front call to start the DDE application.
- In Microsoft Office 2010, the use of DDE is disabled by default. You need to uncheck **Ignore other applications that use Dynamic Data Exchange(DDE)** in advanced options, otherwise DDEConnect will fail.

## DDEExecute

### Purpose:

This function executes a DDE command.

### Syntax:

```
CALL ui.Interface.frontCall("WINDDE","DDEExecute",  
    [ program, document, command, encoding ], [ result ] )
```

### Note:

- *program* is the name of the DDE application.
- *document* is the document that is to be used.
- *command* is the command that needs to be executed.
- *encoding* is an optional parameter. It allows to force the encoding to use between ASCII and wide char/unicode. When not specified, WinDDE will try to retrieve the correct encoding by itself. Possible values are: "UNICODE", "ASCII"
- Refer to the *program* documentation to know the syntax of *command*.
- *result* is an integer variable receiving the status.
- *result* is TRUE if the function succeeded, FALSE otherwise.
-

If the function failed, use DDEError to get the description of the error.

**Warnings:**

- The DDE connection must be opened see DDEConnect.

**DDEFinish**

**Purpose:**

This function closes a DDE connection.

**Syntax:**

```
CALL ui.Interface.frontCall("WINDDE","DDEFinish",  
    [ program, document ], [ result ] )
```

**Note:**

- *program* is the name of the DDE application.
- *document* is the document that is to be closed.
- *result* is an integer variable receiving the status.
- *result* is TRUE if the function succeeded, FALSE otherwise.
- If the function failed, use DDEError to get the description of the error.

**Warnings:**

- The DDE connection must be opened, see DDEConnect.

**DDEFinishAll**

**Purpose:**

This function closes all DDE connections, as well as the DDE server program.

**Syntax:**

```
CALL ui.Interface.frontCall("WINDDE","DDEFinishAll", [], [ result ] )
```

**Note:**

- Closes all DDE connections.
- *result* is an integer variable receiving the status.
- *result* is TRUE if the function succeeded, FALSE otherwise.

**DDEError**

**Purpose:**

This function returns the error information about the last DDE operation.

**Syntax:**

```
CALL ui.Interface.frontCall("WINDDE","DDEError", [], [ errmsg ] )
```

**Note:**

- *errmsg* is the error message. It is set to NULL if no error occurred.

**DDEPeek**

**Purpose:**

This function retrieves data from the specified program and document using the DDE channel.

**Syntax:**

```
CALL ui.Interface.frontCall("WINDDE","DDEPeek",  
    [ program, container, cells, encoding ], [ result, value ] )
```

**Note:**

- *program* is the name of the DDE application.
- *container* is the document or sub-document that is to be used. A sub-document can, for example, be a sheet in Microsoft Excel.
- *cells* represents the working items; see the *program* documentation to know the format of *cells*.
- *encoding* is an optional parameter. It allows to force the encoding to use between ASCII and wide char/unicode. When not specified, WinDDE will try to retrieve the correct encoding by itself. Possible values are: "UNICODE", "ASCII"
- *value* represents the data to be retrieved; see the *program* documentation to know the format of *values*.
- *result* is an integer variable receiving the status.
- *result* is TRUE if the function succeeded, FALSE otherwise.
- If the function failed, use DDEError to get the description of the error.
- *value* is a variable receiving the cells values.

**Warnings:**

- The DDE connection must be opened; see DDEConnect.
- DDEError can only be called once to check if an error occurred.

## DDEPoke

**Purpose:**

This function sends data to the specified program and document using the DDE channel.

**Syntax:**

```
CALL ui.Interface.frontCall("WINDDE","DDEPoke",  
    [ program, container, cells, values, encoding ], [result] )
```

**Note:**

- *program* is the name of the DDE application.
- *container* is the document or sub-document that is to be used. A sub-document can, for example, be a sheet in Microsoft Excel.
- *cells* represents the working items; see the *program* documentation to know the format of *cells*.
- *values* represents the data to be sent; see the *program* documentation to know the format of *values*.
- *encoding* is an optional parameter. It allows to force the encoding to use between ASCII and wide char/unicode. When not specified, WinDDE will try to retrieve the correct encoding by itself. Possible values are: "UNICODE", "ASCII"
- *result* is an integer variable receiving the status.
- *result* is TRUE if the function succeeded, FALSE otherwise.
- If the function failed, use DDEError to get the description of the error.

**Warnings:**

- The DDE connection must be opened; see DDEConnect.
- An error may occur if you try to set many (thousands of) cells in a single operation.

## WinDDE example

### dde\_example.per

```

DATABASE formonly
SCREEN
{
Value to be given to top-left corner :
[f00                               ]
Value found on top-left corner :
[f01                               ]
}
ATTRIBUTES
  f00 = formonly.val;
  f01 = formonly.rval, NOENTRY;

```

### dde\_example.4gl

```

MAIN
-- Excel must be open beforehand
CONSTANT file = "Sheet1"
CONSTANT prog = "EXCEL"
DEFINE val, rval STRING
DEFINE res INTEGER
OPEN WINDOW w1 AT 1,1 WITH FORM "dde_example.per"
INPUT BY NAME val
CALL ui.Interface.frontCall("WINDDE","DDEConnect", [prog,file], [res] )
CALL checkError(res)
CALL ui.Interface.frontCall("WINDDE","DDEPoke", [prog,file,"R1C1",val], [res] );
CALL checkError(res)
CALL ui.Interface.frontCall("WINDDE","DDEPeek", [prog,file,"R1C1"], [res,rval] );
CALL checkError(res)
DISPLAY BY NAME rval
INPUT BY NAME val WITHOUT DEFAULTS
CALL ui.Interface.frontCall("WINDDE","DDEExecute", [prog,file,"[save]"], [res] );
CALL checkError(res)
CALL ui.Interface.frontCall("WINDDE","DDEFinish", [prog,file], [res] );
CALL checkError(res)
CALL ui.Interface.frontCall("WINDDE","DDEFinishAll", [], [res] );
CALL checkError(res)
CLOSE WINDOW w1
END MAIN

FUNCTION checkError(res)
  DEFINE res INTEGER
  DEFINE mess STRING
  IF res THEN RETURN END IF
  DISPLAY "DDE Error:"
  CALL ui.Interface.frontCall("WINDDE","DDEError", [], [mess]);
  DISPLAY mess
  CALL ui.Interface.frontCall("WINDDE","DDEFinishAll", [], [res] );
  DISPLAY "Exit with DDE Error."
  EXIT PROGRAM (-1)
END FUNCTION

```

---

## Windows COM Support

COM stands for Component Object Model. It allows anyone to directly access Windows Applications Objects. You can create instances of those objects, call methods on them, and get or set their properties.



## Using the COM API

With COM Support, you can invoke a Windows application and send or receive data to or from it. To use this functionality, the program must use the Windows Front End.

## The COM API function list

The COM API is based on the front call technique as described in Front End Functions. All COM functions are grouped in the **WinCOM** front end function module.

Table 7-4. COM API functions

Function name	Description
CreateInstance	This function creates an instance of a registered COM object.
CallMethod	This function calls a method on a specified object.
GetProperty	This function gets a property of an object.
SetProperty	This function sets a property of an object.
GetError	This function gets a description of the last error which occurred.
ReleaseInstance	This function releases an Instance (also frees memory).

## Supported syntax

COM language syntax is very flexible and allows lots of notation. Genero WinCOM API is slightly more strict:

- := notation **is allowed** in version 2.00.1e (or later) **only**; for instance:  
`myFunction(SourceType:=3)`
- "no parenthesis" notation **is not allowed**; for instance: `myFunction 3` must be written `myFunction(3)`
- numeric constants **are allowed** in version 2.00.1e (or later) **only**. The constant list depends on the application used via WinCOM, therefore the list is configurable: a file named `etc/WinCOM.cst` gathers all the constants provided today by Microsoft for Office XP. It can be modified to add user-defined constants. Example with Word: `CALL ui.Interface.frontCall("WINCOM","SetProperty", [wdapp,"Selection.Font.Bold","9999998"],[wddoc])` Here, "9999998" stands for the constant **wdToggle** (see `etc/WinCOM.cst`).
- There is no way to handle an array as a method argument. This is also due to 4GL limitation: you can't pass 4GL Arrays to frontcalls.

### CreateInstance

**Purpose:**

This function creates an instance of a registered COM object.

**Syntax:**

```
CALL ui.Interface.frontCall("WinCOM","CreateInstance",  
    [ program ], [handle] )
```

**Note:**

- *program* is the classname of the registered COM object.
- *handle* is an integer variable receiving the status.
- *handle* is -1 if there as an error, otherwise an integer value that can be used for a later call to the API.
- If the function failed, use `GetError` to get the description of the error.

## CallMethod

### Purpose:

This function calls a method on a specified object.

### Syntax:

```
CALL ui.Interface.frontCall("WINCOM","CallMethod",
    [ handle, method, arg1, ... ], [result] ) OR [ handle,
    method(arg1, ...) ], [result] )
```

### Note:

- *handle* is the handle returned by another frontcall (`CreateInstance`, `CallMethod`, `GetProperty`).
- *method* is the member name to call.
- *arg1* (and ...) are the arguments to pass to the method call. Depending on the syntax allowed by the version of the program you're interacting with, arguments might be used inside brackets or outside. The best way for Microsoft applications (such as Excel or Word) is to initially test your code with a macro of the manipulation you're expecting to do. According to the method which is used, arguments may or may not be optional.
- *result* is either a handle or a value of a predefined type.
- *result* is -1 in case of error (use `GetError` to get the description of the error).

## GetProperty

### Purpose:

This function gets a property of an object.

### Syntax:

```
CALL ui.Interface.frontCall("WINCOM","GetProperty",
    [ handle, member ], [result] )
```

### Note:

- *handle* is the handle returned by another frontcall (`CreateInstance`, `CallMethod`, `GetProperty`).
- *member* is the member property name to get.
- *result* is either a handle or a value of a predefined type.
- *result* is -1 in case of error (use `GetError` to get the description of the error).

## SetProperty

### Purpose:

This function sets a property of an object.

### Syntax:

```
CALL ui.Interface.frontCall("WINCOM","SetProperty",
    [handle, member, value], [result] )
```

**Note:**

- *handle* is the handle returned by another frontcall (CreateInstance, CallMethod, GetProperty).
- *member* is the member property name to set.
- *value* is the value to which the property will be set.
- *result* is -1 in case of error (use GetError to get the description of the error), otherwise it is 0.

**GetError****Purpose:**

This function gets a description of the last error which occurred.

**Syntax:**

```
CALL ui.Interface.frontCall("WINCOM","GetError", [], [result] )
```

**Note:**

- *result* is the description of the last error.
- the returned value is NULL if there was no error.

**ReleaseInstance****Purpose:**

This function releases an Instance of a COM object.

**Syntax:**

```
CALL ui.Interface.frontCall("WINCOM","ReleaseInstance", [handle], [result] )
```

**Note:**

- *handle* is the handle returned by another frontcall (CreateInstance, CallMethod, GetProperty).
- *result* is -1 in case of error (use GetError to get the description of the error), otherwise it is 0.

**WinCOM examples**

These topics provide various WinCOM examples.

**Wincom and Excel example**

This example puts "foo" in the first row of the 1st column of an Excel Sheet.

```
DEFINE xlapp INTEGER
DEFINE xlwb INTEGER
MAIN
  DEFINE result INTEGER
  DEFINE str STRING
--initialization of global variables
  LET xlapp = -1
  LET xlwb = -1
--first, we must create an Instance of an Excel Application
  CALL ui.Interface.frontCall("WinCOM", "CreateInstance",
    ["Excel.Application"], [xlapp])
  CALL CheckError(xlapp, __LINE__)
--then adding a Workbook to the current document
  CALL ui.interface.frontCall("WinCOM", "CallMethod",
    [xlapp, "WorkBooks.Add"], [xlwb])
  CALL CheckError(xlwb, __LINE__)
--then, setting it to be visible
```

```

CALL ui.interface.frontCall("WinCOM", "SetProperty",
    [xlapp, "Visible", true], [result])
CALL CheckError(result, __LINE__)
--then CALL SetProperty to set the value of the cell
CALL ui.Interface.frontCall("WinCOM", "SetProperty",
    [xlwb, 'activesheet.Range("A1").Value', "foo"], [result])
CALL CheckError(result, __LINE__)
--then CALL GetProperty to check the value again
CALL ui.Interface.frontCall("WinCOM", "GetProperty",
    [xlwb, 'activesheet.Range("A1").Value'], [str])
CALL CheckError(str, __LINE__)
DISPLAY "content of the cell is: " || str
--then Free the memory on the client side
CALL freeMemory()
END MAIN
FUNCTION freeMemory()
    DEFINE res INTEGER
    IF xlwb != -1 THEN
        CALL ui.Interface.frontCall("WinCOM", "ReleaseInstance", [xlwb], [res] )
    END IF
    IF xlapp != -1 THEN
        CALL ui.Interface.frontCall("WinCOM", "ReleaseInstance", [xlapp], [res] )
    END IF
END FUNCTION
FUNCTION checkError(res, lin)
    DEFINE res INTEGER
    DEFINE lin INTEGER
    DEFINE mess STRING
    IF res = -1 THEN
        DISPLAY "COM Error for call at line:", lin
        CALL ui.Interface.frontCall("WinCOM", "GetError", [], [mess])
        DISPLAY mess
    --let's release the memory on the GDC side
        CALL freeMemory()
        DISPLAY "Exit with COM Error."
        EXIT PROGRAM (-1)
    END IF
END FUNCTION

```

## Wincom and Word example

This example puts "This is a title" centered on the page, underlined, and in bold.

```

DEFINE wdapp INTEGER
DEFINE wddoc INTEGER
MAIN
    DEFINE result INTEGER
    --initialization of global variables
    LET wdapp = -1
    LET wddoc = -1
    --first, we must create an Instance of a Word Application
    CALL ui.Interface.frontCall("WINCOM", "CreateInstance",
        ["Word.Application"], [wdapp])
    CALL CheckError(wdapp, __LINE__)
    --then adding a document
    CALL ui.Interface.frontCall("WINCOM", "CallMethod",
        [wdapp, "Documents.Add"], [wddoc])
    CALL CheckError(wddoc, __LINE__)
    --then, setting it to be visible
    CALL ui.Interface.frontCall("WINCOM", "SetProperty",
        [wdapp, "Visible", true], [result])
    CALL CheckError(result, __LINE__)
    --Centering the cursor for the title
    CALL ui.Interface.frontCall("WINCOM", "SetProperty",
        [wdapp, "Selection.ParagraphFormat.Alignment", "1"], [wddoc])
    CALL CheckError(wddoc, __LINE__)
    --Underlining the title
    CALL ui.Interface.frontCall("WINCOM", "SetProperty",

```

```

        [wdapp,"Selection.Font.Underline","1"],[wddoc])
    CALL CheckError(wddoc, __LINE__)
--Putting the title in bold
    CALL ui.Interface.frontCall("WINCOM","SetProperty",
        [wdapp,"Selection.Font.Bold","9999998"],[wddoc])
    CALL CheckError(wddoc, __LINE__)
--Typing the title's text
    CALL ui.Interface.frontCall("WINCOM","CallMethod",
        [wdapp,'Selection.TypeText("This is a title)']],[wddoc])
    CALL CheckError(wddoc, __LINE__)
--then Free the memory on the client side
    CALL freeMemory()
END MAIN
FUNCTION freeMemory()
    DEFINE res INTEGER
    IF wddoc != -1 THEN
        CALL ui.Interface.frontCall("WinCOM","ReleaseInstance", [wddoc], [res] )
    END IF
    IF wdapp != -1 THEN
        CALL ui.Interface.frontCall("WinCOM","ReleaseInstance", [wdapp], [res] )
    END IF
END FUNCTION
FUNCTION checkError(res, lin)
    DEFINE res INTEGER
    DEFINE lin INTEGER
    DEFINE mess STRING
    IF res = -1 THEN
        DISPLAY "COM Error for call at line:", lin
        CALL ui.Interface.frontCall("WinCOM","GetError",[],[mess])
        DISPLAY mess
--let's release the memory on the GDC side
        CALL freeMemory()
        DISPLAY "Exit with COM Error."
        EXIT PROGRAM (-1)
    END IF
END FUNCTION

```

## Wincom and Outlook example

The following example executes Outlook, creates a new contact, and saves it in your contact list.

```

DEFINE outapp INTEGER
DEFINE outit INTEGER
DEFINE outcon INTEGER
DEFINE outsav INTEGER
MAIN
    DEFINE result INTEGER
    DEFINE str STRING
--initialization of global variables
    LET outapp = -1
    LET outit = -1
    LET outcon = -1
    LET outsav = -1
--first, we must create an Instance of an Outlook Application
    CALL ui.interface.frontcall("WinCOM", "CreateInstance",
        ["Outlook.Application"], [outapp])
    CALL CheckError(outapp, __LINE__)
--then, creating a contact object
    CALL ui.interface.frontcall("WinCOM", "CallMethod",
        [outapp, "CreateItem(o1ContactItem)"], [outit])
    CALL CheckError(outit, __LINE__)
--then, displaying the contact form
    CALL ui.interface.frontCall("WinCOM", "CallMethod",
        [outit, "Display"], [outcon])
    CALL CheckError(outcon, __LINE__)
--CALL SetProperty to fill the various fields with the values you expect
#First Name

```

```

CALL ui.interface.frontCall("WinCOM", "SetProperty",
  [outit, "FirstName", "Lionel"], [result])
CALL CheckError(result, __LINE__)
#1st email address
CALL ui.interface.frontCall("WinCOM", "SetProperty",
  [outit, "EmailAddress", "lif@4js.com"], [result])
CALL CheckError(result, __LINE__)
#Business address
CALL ui.interface.frontCall("WinCOM", "SetProperty",
  [outit, "BusinessAddress", "1 rue de Berne"], [result])
CALL CheckError(result, __LINE__)
--then, CALL GetProperty to check the values again
CALL ui.Interface.frontCall("WinCOM", "GetProperty",
  [outit, "FirstName"], [str])
CALL CheckError(str, __LINE__)
DISPLAY "First Name of the new contact is " || str
CALL ui.Interface.frontCall("WinCOM", "GetProperty",
  [outit, "EmailAddress"], [str])
CALL CheckError(str, __LINE__)
DISPLAY "1st email of the new contact is " || str
CALL ui.Interface.frontCall("WinCOM", "GetProperty",
  [outit, "BusinessAddress"], [str])
CALL CheckError(str, __LINE__)
DISPLAY "Business Address of the new contact is " || str
--at the end, saving the contact
CALL ui.interface.frontCall("WinCOM", "CallMethod", [outit, "Save"], [outsav])
CALL CheckError(outsav, __LINE__)
--then Free the memory on the client side
CALL freeMemory()
END MAIN
FUNCTION freeMemory()
  DEFINE res INTEGER
  IF outit != -1 THEN
    CALL ui.Interface.frontCall("WinCOM","ReleaseInstance", [outit], [res] )
  END IF
  IF outapp != -1 THEN
    CALL ui.Interface.frontCall("WinCOM","ReleaseInstance", [outapp], [res] )
  END IF
END FUNCTION
FUNCTION checkError(res, lin)
  DEFINE res INTEGER
  DEFINE lin INTEGER
  DEFINE mess STRING
  IF res = -1 THEN
    DISPLAY "COM Error for call at line:", lin
    CALL ui.Interface.frontCall("WinCOM","GetError",[],[mess])
    DISPLAY mess
  --let's release the memory on the GDC side
  CALL freeMemory()
  DISPLAY "Exit with COM Error."
  EXIT PROGRAM (-1)
  END IF
END FUNCTION

```

**Tip:** You may find the various Outlook objects (such as *ContactItem* object), methods (such as the *CreateItem* method), and properties (such as the *FirstName* or *BusinessAddress* properties) on the Microsoft Developer Network.

## Wincom and Internet Explorer example

The following example executes Internet Explorer on a defined URL with the address bar masked.

```

DEFINE ieapp INTEGER
DEFINE ienav INTEGER
MAIN
  DEFINE result INTEGER

```

```

--initialization of global variables
  LET ieapp = -1
  LET ienav = -1
--first, we must create an Instance of Internet Explorer application
  CALL ui.Interface.frontCall("WinCOM", "CreateInstance",
  ["InternetExplorer.Application"], [ieapp])
  CALL CheckError(ieapp, __LINE__)
--then, specifying the URL you want to load
  CALL call ui.interface.frontCall("WinCOM", "CallMethod",
  [ie_app, "Navigate", "www.4js.com"], [ienav])
  CALL CheckError(ienav, __LINE__)
--then, masking the address bar
  CALL ui.interface.frontCall("WinCOM", "SetProperty",
  [ieapp, "AddressBar", false], [result])
  CALL CheckError(result, __LINE__)
--then, setting it to visible
  CALL ui.interface.frontCall("WinCOM", "SetProperty", [ieapp, "Visible", true],
  [result])
  CALL CheckError(result, __LINE__)
--then Free the memory on the client side
  CALL freeMemory()
END MAIN
FUNCTION freeMemory()
  DEFINE res INTEGER
  IF ienav != -1 THEN
    CALL ui.Interface.frontCall("WinCOM", "ReleaseInstance", [ienav], [res] )
  END IF
  IF ieapp != -1 THEN
    CALL ui.Interface.frontCall("WinCOM", "ReleaseInstance", [ieapp], [res] )
  END IF
END FUNCTION
FUNCTION checkError(res, lin)
  DEFINE res INTEGER
  DEFINE lin INTEGER
  DEFINE mess STRING
  IF res = -1 THEN
    DISPLAY "COM Error for call at line:", lin
    CALL ui.Interface.frontCall("WinCOM", "GetError", [], [mess])
    DISPLAY mess
--let's release the memory on the GDC side
    CALL freeMemory()
    DISPLAY "Exit with COM Error."
    EXIT PROGRAM (-1)
  END IF
END FUNCTION

```

---

## Windows Mail extension

### Send mail using MAPI

**MAPI** is an acronym for Messaging Application Programming Interface. The **MAPI** extension will create a new mail in the default mailer software, which needs to be "MAPI" compatible, and ask the user to send the mail. The mail sent using MAPI will be stored by the default mailer software in the same way as any other mail created by the user.

### Send mail using an SMTP server

Another method of sending mail is to connect directly to an **SMTP** server (Simple Mail Transfer Protocol is the de facto standard for email transmission across the Internet). The extension will connect to a given **SMTP** server and send the mail through this server. The mail is not kept on the client side.

## The WinMail API

The WinMail API is based on the front call technique as described in Front End Functions. All WinMail functions are grouped in the **WinMail** front end function module.

Table 7-5. WinMail API functions: General

Function name	Description
Init	This function prepares a message
Close	This function closes the message
SetBody	This function sets the body of the mail
SetSubject	This function sets the subject of the mail
AddTo	This function adds a "To" addressee
AddCC	This function adds a "CC" addressee
AddBCC	This function adds a "BCC" addressee
AddAttachment	This function adds an attachment
SendMail	This function sends the mail
GetError	This function retrieves the last error message

The following functions are needed when you use SMTP server connections:

Table 7-6. WinMail AIP functions: SMTP-specific

Function name	Description
setSmtp	This function sets the SMTP server to use
setFrom	This function sets the sender address

### Init

#### Purpose:

This function initializes the module. It returns the identifier for the message, which will be used in other functions.

#### Syntax:

```
CALL ui.Interface.frontCall("WinMail","Init", [], [id ] )
```

#### Note:

- *ret* is the identifier of the message initialized.
- For each Init function, a Close must be called.

### Close

#### Purpose:

This function clears all information corresponding to a message, and frees the memory occupied by the message.

#### Syntax:

```
CALL ui.Interface.frontCall("WinMail","Close", [id], [ result ] )
```

#### Note:

- *id* is the message identifier.



- *result* is the status of the function.

## **SetBody**

### **Purpose:**

This function sets the body of the mail.

### **Syntax:**

```
CALL ui.Interface.frontCall("WinMail","SetBody", [ id, body ], [ result ] )
```

### **Note:**

- *id* is the message identifier.
- *body* is the string text containing the body of the mail.
- *result* is the status of the function.

## **SetSubject**

### **Purpose:**

This function sets the subject of the mail.

### **Syntax:**

```
CALL ui.Interface.frontCall("WinMail","SetSubject", [ id, subject ], [ result ] )
```

### **Note:**

- *id* is the message identifier.
- *subject* is the string text containing the subject of the mail.
- *result* is the status of the function.

## **AddTo, AddCC, AddBCC**

### **Purpose:**

These functions add a "To" Addressee to the mail. The addressee can be in one of the following categories:

- "To" (to)
- "CC" (carbon copy)
- "BCC" (blind carbon copy)

The Addressee has a name and a mail address.

### **Syntax:**

```
CALL ui.Interface.frontCall("WinMail","AddTo", [ id, name, address ], [ result ] )  
CALL ui.Interface.frontCall("WinMail","AddCC", [ id, name, address ], [ result ] )  
CALL ui.Interface.frontCall("WinMail","AddBCC", [ id, name, address ], [ result ] )
```

### **Note:**

- *id* is the message identifier.
- *name* is the name to be displayed in the mail.
- *address* is the mail address to be used for this addressee.
- *result* is the status of the function.

## **AddAttachment**

### **Purpose:**

This function adds a file as an attachment to the mail. The file must be located on the front-end.

**Syntax:**

```
CALL ui.Interface.frontCall("WinMail","AddAttachment", [ id, fileName ], [ result ] )
```

**Note:**

- *id* is the message identifier.
- *fileName* is the path of the attachment; the path can be relative to the directory from which GDC is run, or absolute.
- *result* is the status of the function.

## SendMail

**Purpose:**

This function sends the mail. Using SMTP, the SMTP server is directly used. Using MAPI, the default mailer software is called to create the mail. The user must press the "send" button to send the mail.

**Syntax with SMTP:**

```
CALL ui.Interface.frontCall("WinMail","SendMailSMTP", [ id ], [ result ] )
```

**Syntax with MAPI:**

```
CALL ui.Interface.frontCall("WinMail","SendMailMAPI", [ id ], [ result ] )
```

**Note:**

- *id* is the message identifier.
- *result* is TRUE in case of success; use GetError to get the description of the error when needed.

**Important:**

- MAPI needs to log in to the mailer software. The first login could take time, depending on the mailer software. Your Genero application will be blocked until MAPI returns.
- MAPI depends on the mailer software for error management. For instance, Mozilla Thunderbird returns "success" when the mail is created, but Outlook 2002 only returns "success" when the mail is sent.

## GetError

**Purpose:**

This function gets a description of the last error that occurred.

**Syntax:**

```
CALL ui.Interface.frontCall("WinMail","GetError", [ id ], [ result ] )
```

**Note:**

- *id* is the message identifier.
- *result* is the description of the last error.
- the returned value is NULL if there was no error.

## SetSmtp

### Purpose:

This function sets the SMTP server to be used.

### Syntax:

```
CALL ui.Interface.frontCall("WinMail","SetSmtp", [ id, smtp:port ], [ result ] )
```

### Note:

- *id* is the message identifier.
- *smtp* is the string text containing the SMTP server to be used.
- *port* is optional. It allows to specify a port for your SMTP server. When not specified, the default port remains 25.
- *result* is the status of the function.

## SetFrom

### Purpose:

This function sets sender information. (This is needed for SMTP connections).

### Syntax:

```
CALL ui.Interface.frontCall("WinMail","SetFrom", [ id, name, address ], [ result ] )
```

### Note:

- *id* is the message identifier.
- *name* is the name to be displayed in the mail.
- *address* is the mail address to be used for this addressee.
- *result* is the status of the function.

## WinMail examples

These topics provide WinMail examples.

### Mail using MAPI

The following example sends a mail using MAPI.

```
MAIN
  DEFINE result, id INTEGER
  DEFINE str STRING
  -- first, we initialize the module
  CALL ui.Interface.frontCall("WinMail", "Init", [], [id])
  -- Set the body of the mail
  CALL ui.interface.frontCall("WinMail", "SetBody",
    [id, "This is a text mail using WinMail API - MAPI"], [result])

  -- Set the subject of the mail
  CALL ui.interface.frontCall("WinMail", "SetSubject",
    [id, "test mail - ignore it"], [result])
  -- Add an Addressee as "TO"
  CALL ui.Interface.frontCall("WinMail", "AddTo",
    [id, "myBoss", "boss@mycompany.com"], [result])
  -- Add another Addressee as "BCC"
  CALL ui.Interface.frontCall("WinMail", "AddBCC",
    [id, "my friend", "friend@mycompany.com"], [result])
  -- Add Two attachments
  CALL ui.Interface.frontCall("WinMail", "AddAttachment",
    [id, "c:\\mydocs\\report.doc"], [result])
  CALL ui.Interface.frontCall("WinMail", "AddAttachment",
    [id, "c:\\mydocs\\demo.png"], [result])
```

```

-- Send the mail via the default mailer
CALL ui.Interface.frontCall("WinMail", "SendMailMAPI", [id], [result])
IF result == TRUE THEN
  DISPLAY "Message sent succesfully"
ELSE
  CALL ui.Interface.frontCall("WinMail", "GetError", [id], [str])
  DISPLAY str
END IF
CALL ui.Interface.frontCall("WinMail", "Close", [id], [result])
END MAIN

```

## Mail using SMTP server

The following example sends a mail using an SMTP server:

```

MAIN
  DEFINE result, id INTEGER
  DEFINE str STRING

-- first, we initialize the module
  CALL ui.Interface.frontCall("WinMail", "Init", [], [id])

-- Set the body of the mail
  CALL ui.interface.frontCall("WinMail", "SetBody", [id,
    "This is a text mail using WinMail API - MAPI"], [result])

-- Set the subject of the mail
  CALL ui.interface.frontCall("WinMail", "SetSubject", [id,
    "test mail - ignore it"], [result])

-- Set the mail sender
  CALL ui.Interface.frontCall("WinMail", "SetFrom", [id, "mySelf",
    "me@mycompany.com"], [result])

-- Set the SMTP server
  CALL ui.Interface.frontCall("WinMail", "SetSmtP", [id, "smtp.mycompany.com"],
    [result])

-- Add an Addressee as "TO"
  CALL ui.Interface.frontCall("WinMail", "AddTo", [id, "myBoss",
    "boss@mycompany.com"], [result])

-- Add another Addressee as "BCC"
  CALL ui.Interface.frontCall("WinMail", "AddBCC", [id, "my friend",
    "friend@mycompany.com"], [result])

-- Add Two attachments
  CALL ui.Interface.frontCall("WinMail", "AddAttachment", [id,
    "c:\mydocs\report.doc"], [result])
  CALL ui.Interface.frontCall("WinMail", "AddAttachment", [id,
    "c:\mydocs\demo.png"], [result])

-- Send the mail via smtp
  CALL ui.Interface.frontCall("WinMail", "SendMailSMTP", [id], [result])
  IF result == TRUE THEN
    DISPLAY "Message sent succesfully"
  ELSE
    CALL ui.Interface.frontCall("WinMail", "GetError", [id], [str])
    DISPLAY str
  END IF
  CALL ui.Interface.frontCall("WinMail", "Close", [id], [result])
END MAIN

```

---

## Chapter 8. General terms

This documentation uses several terms that must be clarified for a good understanding. Here is a short description for all these terms:

### **Product**

The *Product* defines all software components that compose the information system managing a given domain. Usually, the domains covered by programs written in BDL are business oriented.

### **End User**

The *End User* is the person that uses the Product; that person works on hardware called the Workstation.

### **Programs**

The *Programs* are the software components that are developed and distributed by the supplier of the Product. *Programs* typically implement business rules and processing, usually called Business Logic. *Programs* are executed by the Runtime System on the Application Server machine. These components are typically p-code modules, forms and additional files.

### **Developer**

The *Developer* is the person in charge of the conception and implementation of the Product components.

### **Application Data**

*Application Data* defines the data manipulated by the Product. It is typically managed by one or more Database Systems. The *Application Data* has a volatile state when loaded in the Runtime System, and it has a static state when stored in the Database System.

### **Database**

The *Database* is a logical entity regrouping the Application Data. It is managed by the Database System.

### **Database System**

The *Database System* is the software that manages data storage and searching; it is usually installed on the Database Server machine and is supported by a tier software vendor. It is the software managing the Data in the Three-Tier C/S model.

### **Development Database**

The *Development Database* is the Database used in the application development environment.

### **Production Database**

The *Production Database* is the Database used on production sites.

### **Front End**

The *Front End* is the software that manages the display of the User Interface on the Workstation machine. This component is historically called "The Client", in a thin Client/Server context. It is the software managing the Presentation in the Three-Tier C/S model.

### **Runtime System**

The *Runtime System* is the software that manages the execution of the Programs, where the Business Logic is processed. It is typically

implemented by the *Dynamic Virtual Machine* (DVM) and historically called "The Runner". It is the software managing the Processing in the Three-Tier C/S model.

**User Interface**

The *User Interface* defines the parts of the Programs that interact with the end user, including interactive elements like windows, screens, input fields, buttons and menus. It is displayed on the Workstation. This can typically be implemented by different kinds of Front Ends, based on ASCII terminals, graphical platforms (Microsoft Windows, X11) or even through web protocols like HTML over HTTP.

**Workstation**

The *Workstation* identifies the hardware used by the End User to interact with the Product. It can be an ASCII Terminal, a PC, a diskless station or even a cellular phone, as long as a Front End is available on that hardware.

---

## Chapter 9. Security terms

The security section of the documentation uses several terms that must be clarified for a good understanding. Here is a short description for all these terms:

### **Firewall Router**

This is a device that isolates the corporate network from the Internet. It typically allows connections to the Internet, but also prevents connections from entering. They can usually be configured to allow/prevent several conditions. They can be configured to allow a port connection from the Internet to go through to a machine. This can be done either by allowing the connection straight through or translating it to a different port.

**NAT** *Network Address Translation* is a method of allowing computers to access the Internet without having them be assigned real Internet addresses. The connections must originate from the internal machines to reach Internet addresses. The NAT router will then put these on the Internet using the router's IP address. When data is returned it forwards the data to the requesting internal machine. Part of this process includes mapping what internal IP/Port combinations correspond to external port usage. Doing so allows the router to know where data needs to be sent when it returns. Special port mappings can be made to specific internal IP addresses to support connections originating from the Internet. Other configurable values might be session timers that will be explored in the section.

### **Private Network**

This is the network used in the corporation that is private and trusted. Most companies tightly control what is plugged in so they can ensure the data is safe.

**VPN** *Virtual Private Network* is a method of tunnelling through an existing connection back to the corporate LAN. It provides end-to-end encrypted connections. These types of connections are usually equivalent to being plugged into the office LAN.

### **Encryption of all Data**

Genero requires a TCP connection for the GUI data transmission. If the GDC short cuts are being used there is also a connection needed to start the application that may require a log in. Both connections in this case are encrypted.

### **Password/login Encrypted**

Genero logs in and executes an application when the short cuts are used. This connection is encrypted. The connection carrying the GUI data is not encrypted.

### **Keep Alive**

Typical TCP connections don't cause any network traffic when idle unless the KeepAlive flag is set. This flag will prevent the session from timing out and thus prevent the session from closing. This also assumes that the firewalls don't expire the session during the keep alive interval.

### **Port Forwarding**

The method referred to is implemented in the Secure Shell (ssh). The ssh can be told to listen to a port and tunnel it through an existing ssh session

and present it to a port on the other machine. This method is used to listen to a port on the server side and direct the data to the GDC on the client side.

**Note:** This document covers system configuration using the following environment:

- Genero Desktop Client Release 1.20.1a (under Windows, Linux and Mac Os 10)
- Genero DVM Release 1.20.1a (Under Linux and Windows)
- Different Openssh Server 3.x.yy under Linux



---

## Appendix. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

---

### Accessibility features for IBM Informix products

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

#### Accessibility features

The following list includes the major accessibility features in IBM Informix products. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

#### Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

#### Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

#### IBM and accessibility

See the *IBM Accessibility Center* at <http://www.ibm.com/able> for more information about the IBM commitment to accessibility.



---

## Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM web sites are provided for convenience only and do not in any manner serve as an endorsement of those web sites. The materials at those web sites are not part of the materials for this IBM product and use of those web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, any such examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com)<sup>®</sup> are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/us/en/copytrade.shtml](http://www.ibm.com/legal/us/en/copytrade.shtml).

Four Js is a registered trademark of Four Js Development Tools Ltd.

Genero and its logo are registered trademarks of Four Js Development Tools Europe Ltd.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.







Product Number: 5725-D13

Printed in USA

SC27-3854-01

