Genero Desktop Client User Guide
## Contents

GDC User Guide................................................................. 4

What's New in 3.10............................................................ 5

### General

What is the Genero Desktop Client........................................... 5
Compatibility........................................................................ 6
Starting and Configuring the GDC........................................... 6
Start the Genero Desktop Client............................................. 6
Configure the Genero Desktop Client..................................... 7
Apply an additional configuration file....................................... 18
GDC configuration file directories.......................................... 18
Microsoft® Windows® Security Blocking............................... 19

### Applications

The Shortcut System......................................................... 19
Creating Shortcuts using the Shortcut Wizard.......................... 20
Shortcut Management....................................................... 28
Shortcuts and environment variables...................................... 29
Customize your Login Box.................................................. 29
Connections Panel............................................................ 31
The Terminals Panel......................................................... 33
The Debug Panel and the logging system............................... 35
Using the logging system.................................................... 35
Viewing the Debug Console................................................. 36

### Features

The Command Line........................................................... 38
Command line options....................................................... 38
Command line examples.................................................... 41
Inspecting the AUI tree..................................................... 42
Printing a screen shot....................................................... 42
Local actions.................................................................... 43
Localization encoding list.................................................. 44
Accessibility....................................................................... 45
Auto Update....................................................................... 45
Debug Web content.......................................................... 48
Spellchecking in Web Components....................................... 49

### Upgrading

New features of the GDC.................................................... 50
GDC 3.10 new features....................................................... 50
GDC 3.00 new features....................................................... 50
GDC 2.50 new features....................................................... 51
Security..............................................................98

GDC 2.40 new features..........................................................52
GDC 2.32 new features..........................................................58
GDC 2.30 new features..........................................................59
GDC 2.22 new features..........................................................70
GDC 2.21 new features..........................................................73
GDC 2.20 new features..........................................................79

Upgrade Guides for the GDC...............................................88
GDC 3.10 upgrade guide.........................................................88
GDC 3.00 upgrade guide.........................................................89
GDC 2.5x upgrade guide.........................................................90
GDC 2.4x upgrade guide.........................................................90
GDC 2.3x upgrade guide.........................................................91
GDC 2.2x upgrade guide.........................................................93

Front End Extensions.........................................................140

Legal notices.........................................................................140

GDC configuration file directories........................................141

Copyrights and Trademarks.................................................142

Index..................................................................................143
# Genero Desktop Client User Guide

Manual organization at a glance.

<table>
<thead>
<tr>
<th>General on page 5</th>
<th>Applications on page 19</th>
<th>Features on page 37</th>
<th>Upgrading on page 49</th>
</tr>
</thead>
<tbody>
<tr>
<td>• GDC 3.10 new features on page 5</td>
<td>• The Shortcut System on page 19</td>
<td>• The Command Line on page 38</td>
<td>• New Features</td>
</tr>
<tr>
<td>• What is the Genero Desktop Client on page 5</td>
<td>• Customize your Login Box on page 29</td>
<td>• Inspecting the AUI tree on page 42</td>
<td>• Upgrade Guides</td>
</tr>
<tr>
<td>• Compatibility on page 6</td>
<td>• Connections Panel on page 31</td>
<td>• Printing a screen shot on page 42</td>
<td></td>
</tr>
<tr>
<td>• Starting and Configuring the GDC on page 6</td>
<td>• The Terminals Panel on page 33</td>
<td>• Local actions on page 43</td>
<td></td>
</tr>
<tr>
<td>• Microsoft Windows Security Blocking on page 19</td>
<td>• The Debug Panel and the logging system on page 35</td>
<td>• Localization encoding list on page 44</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security on page 98</th>
<th>Front End Extensions on page 140</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Security levels on page 98</td>
<td>• Front End Extensions on page 140</td>
</tr>
<tr>
<td>• Security terms on page 100</td>
<td></td>
</tr>
<tr>
<td>• GDC and SSH on page 102</td>
<td></td>
</tr>
<tr>
<td>• GDC and SSH simple setup on page 104</td>
<td></td>
</tr>
<tr>
<td>• Port Forwarding and Firewalls on page 105</td>
<td></td>
</tr>
<tr>
<td>• Implementing a Secure Server with GDC on page 120</td>
<td></td>
</tr>
<tr>
<td>• SSH Configuration Troubleshooting on page 135</td>
<td></td>
</tr>
<tr>
<td>• Microsoft firewall configuration on page 138</td>
<td></td>
</tr>
<tr>
<td>• Microsoft User Account Control on page 139</td>
<td></td>
</tr>
</tbody>
</table>
GDC 3.10 new features

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 3.10.

Corresponding upgrade guide: GDC 3.10 upgrade guide on page 88.

Table 1: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDC uses Qt 5.9. A key benefit of Qt 5.9 is the new Chromium 56 renderer for WebComponents.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The auto-update feature allows users to update an existing Genero Desktop Client installation.</td>
<td>See Auto Update on page 45.</td>
</tr>
<tr>
<td>Debugging of Web Components can take advantage of the Qt WebEngine module, which makes it easy to inspect and debug layout and performance issues of any Web content from a Chrome browser.</td>
<td>See Debug Web content on page 48.</td>
</tr>
<tr>
<td>GDC provides a mechanism for bypassing certificate errors.</td>
<td>See Bypassing certificate errors on page 136.</td>
</tr>
<tr>
<td>GDC provides an interface for deleting stored passwords.</td>
<td>See Security configuration options on page 15.</td>
</tr>
<tr>
<td>The GDC provides spellchecking support in Web Components.</td>
<td>See Spellchecking in Web Components on page 49.</td>
</tr>
<tr>
<td>Drag-and-drop to export and import shortcuts when the monitor is in admin mode.</td>
<td>See Shortcut Management on page 28.</td>
</tr>
<tr>
<td>Clear the Web cache to support development when using webviews.</td>
<td>See Web cache on page 10.</td>
</tr>
</tbody>
</table>

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 on page 5
- Starting and Configuring the GDC on page 6

What is the Genero Desktop Client

The Genero Desktop Client is a graphical front-end for a Genero application.

The Genero Desktop Client is multi-platform and can run under Windows®, macOS™, and Linux®.

For a detailed list of supported operating systems, refer to the System Support matrix (available on the Four Js Web site in the product and documentation download area) or contact your support center. This matrix also informs you which operating systems will no longer be supported as of the next release.
Compatibility

Both the 3.00 and 3.10 versions of the Genero DVM and the Genero Application Server are supported.

Tip: Upgrade to the latest version of Genero Desktop Client to benefit from improvements and fixes, even if running FGL 3.00 or FGL+GAS 3.00.

GDC 3.10 is able to run in direct mode (SSH) with FGL 3.00 or FGL 3.10. See Create a Direct Connection shortcut on page 21.

GDC 3.10 is able to run on the UA protocol using FGL+GAS 3.00 or FGL+GAS 3.10. See Create a HTTP Connection shortcut on page 25.

Starting and Configuring the GDC

Before displaying an application using the Genero Desktop Client as the front-end, you may need to configure and/or start the client.

• Start the Genero Desktop Client on page 6
• Configure the Genero Desktop Client on page 7
• Apply an additional configuration file on page 18

Start the Genero Desktop Client

The method used to start GDC is based on your operating system.

To start the GDC:

• Under Windows® systems, you can use the shortcut on the Start Menu.
• Under X11 systems, you can also use the shortcut on the Start Menu, or 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 macOS™ systems, the installer will add GDC to the Applications folder. You can also create a desktop shortcut to launch GDC from the command line. See Create a Genero Desktop Client desktop shortcut on macOS on page 6.

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.

Create a Genero Desktop Client desktop shortcut on macOS™

Create a desktop shortcut for the Genero Desktop Client (GDC) on the macOS™ operating system.

1. Go to Applications/Utilities > AppleScript > Script Editor.
2. Complete and enter the following command:

```bash
try
  do shell script "~/Users<USERNAME>/Applications/Genero\ Desktop\Client\<VERSIONNUMBER>.app/Contents/MacOS/gdc <COMMANDLINEOPTIONS>"
end try
```

Note: For more information about the modes that you can start the shortcut in, see command line options

Option: to verify that the script is correct, click Run.
3. Save the script on the desktop as an Application bundle.
Configure the Genero Desktop Client

Configure the Genero Desktop Client by accessing the configuration tabs.

Click the Options icon to display the configuration options panel. The Options panel is only available in administration mode.

The configuration options are organized across tabs. After you modify a configuration, select Apply to validate and save your changes or Restore to discard your changes.

- Preferences configuration options on page 7
- Advanced configuration options on page 9
- Connection configuration options on page 13
- Security configuration options on page 15
- Report configuration options on page 17

Related concepts
The Shortcut System on page 19
The Genero Desktop Client (GDC) can store the information needed to start an application. The information is stored as a shortcut.

Preferences configuration options
Use the Preferences tab to configure an images path, an icon path, and font overriding.

![Figure 1: Genero Desktop Client Options; Preferences tab](image)

The Preferences tab allows you to set path and font overriding information.
**Important:** Changes will not be applied until the "Apply" button is clicked.

**Set path options**

These paths can be set:

**Images**

Specifies the path for the 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. If it cannot find the image, it draws a "..." picture.

**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.

**Note:**

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

![Figure 2: Images field with invalid directory entered](image)

Most of the fields have auto completion:

![Figure 3: Images field with autocomplete feature](image)

**Set font overriding options**

These font overriding options can be set:

**Default**

Specifies the default font for GDC. This font will be used everywhere in your applications.

**Monospace**

Specifies the default fixed font for GDC. This font will be used when the fixed font attribute is defined.
**Advanced configuration options**

Use the Advanced tab to configure the caches, the stored settings, dictionary lists, and buttons style.

![Genero Desktop Client Options; Advanced Tab](image)

**Figure 4: Genero Desktop Client Options; Advanced Tab**

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

- Image Cache
- Proxy Cache
- Web cache
- Stored settings
- Dictionary list
- Buttons style
**Image Cache**

The Genero Desktop Client stores images that have been retrieved remotely, such as when the images are retrieved using http (either because the url specifies http, or a PICTURE alias is used with Genero Application Server), or from the runtime system side (using FGLIMAGEPATH).

The size of the image cache can be configured. Images are stored in the AppCacheDir directory (see GDC configuration file directories on page 18). When the cache is full, those images which have not been recently used are removed from the cache.

A memory cache is used by GDC. Images that are frequently used are cached to be loaded as fast as possible.

The imageCache style has been introduced to manage the cache; the disk cache and memory cache. 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.

**Note:** When configuration settings for the image cache in GDC are modified in one monitor, the settings are applied to all monitors for a user. For example, if you delete the image cache in one monitor, it is also deleted in all other monitors.

Note that the image cache is common to all applications for a user. This can result in an error if the cache is enabled and there are image files with the same file name. Take the following example of an error that occurs when two applications try to display two different images with the same file name:

1. The first application will write the first image to the cache.
2. The second application, while trying to load the second image with the same name, will search the cache by file name and load the first image.
3. The second application incorrectly displays the first image.

**Proxy Cache**

The Clear button will clear the cache.

**Web cache**

Web component widgets use webviews. Webviews created by the GDC share a common disk cache, known here as the Web cache.

When developing applications, the Web cache can prevent the developer from seeing changes in the Web component, as old versions of the files saved in the cache may be used instead of downloading the latest versions from the server. To force the webview to download the latest versions, users need to clear the cache.

The Clear button clears the cache.

**Tip:** It is recommended to close any applications displaying a webview prior to clearing the cache, as it will wipe all cache entries from the disk.

**Stored settings**

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.
Dictionary list

The dictionary list relates to the spellchecking feature for Web components. Spellchecking of Web components is based on dictionaries stored in:

- GDCDIR/bin/qtwebengine_dictionaries on Windows® and Linux® systems.
- GDCDIR/Contents/Resources/qtwebengine_dictionaries on macOS™ systems.

The dictionaries are .bdic files. GDC populates the Select Language context menu for spellchecking of Web components based on the list of .bdic files found in this directory. GDC creates this list on startup; click Refresh to update the list based on the latest contents of the directory.

For the .bdic files packaged with GDC, a transposition of the file name is made to display the name of the language. For example, instead of "it_IT" you will see "Italian". For all added .bdic files, the name of the file is used in the context menu.

Buttons style

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 5: Raised buttons with icons
Figure 6: Raised buttons without icons
Connection configuration options
Use the Connection tab to configure HTTP proxy, HTTP retries, and ping events.

Figure 7: Genero Desktop Client Options; Connection Tab

These options can be configured in the Connection panel:

- **Http Proxy**
- **Http Retries**
- **Ping event**
- **Notifications**

**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;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).
Notifications
Enable or disable the display of tray notifications.
Security configuration options
Access the Security tab to configure the security level, clear stored passwords, and view/clear known hosts.
Genero Desktop Client - port: 6400 (any)

Options

<table>
<thead>
<tr>
<th>Preferences</th>
<th>Advanced</th>
<th>Connection</th>
<th>Security</th>
<th>Report</th>
</tr>
</thead>
</table>

Security level

**Security Level 1**

The user is warned when a connection occurs.

Passwords

Manage...

Known Hosts

Manage...
These options can be configured on the **Security** panel:

- **Security Level**
- **Passwords**
- **Known Hosts**

**Security Level**

Use the slide to set the security level. See Security levels on page 98 for more information.

**Passwords**

Clear passwords stored by the GDC.

Click the **Manage** button to open the Manage passwords dialog. From this dialog, you can clear selected (or all) passwords stored by the GDC.

**Known Hosts**

View or remove known hosts.

Click the **Manage** button to open the Manage known hosts dialog. From this dialog, you can view known host details and remove selected (or all) known hosts.

See Bypassing certificate errors on page 136 for more information.

**Report configuration options**

Use the Report tab to configure default printer and font settings for REPORT TO PRINTER behavior.

![Figure 9: Genero Desktop Client Options; Report Tab](image)

Options for the printer and font, used with REPORT TO PRINTER behavior:

- **Ask once**: The Genero Desktop Client will ask for the parameter once, and then keep the choice in memory until the Genero Desktop Client closes.
- **Ask on every report**: The Genero Desktop Client will ask every time a report is printed.
- **Use default**: Use the system default printer or the Genero Desktop Client's default font.
- **Use**: Use a specified printer or font.
Apply an additional configuration file

You can use the --config option to specify an additional configuration file. Configuration settings specified in this file take precedence over the configuration settings defined in the default configuration file.

The Genero Desktop Client stores most configuration options in a user-specific config.xml file (see GDC configuration file directories on page 18 for more information). Use an additional configuration file to override entries in the config.xml file. Specify the additional configuration file with the --config option.

Configuration options are read from:

1. The configuration file specified by the --config option.
2. The default config.xml for options not specified in the specific configuration file.

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

Related concepts
The Command Line on page 38
Using the command line with the Genero Desktop Client.

GDC configuration file directories

The GDC configuration files are stored in two (default) directories: AppDataDir and AppCacheDir.

Table 2: Configuration file directories shows the locations of the default directories for the GDC configuration files.

<table>
<thead>
<tr>
<th>Directory name</th>
<th>Directory location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppDataDir</td>
<td></td>
<td>Contains:</td>
</tr>
<tr>
<td>Windows®</td>
<td>C:\Users &lt;USERNAME&gt;\AppData \Roaming\Four Js\Genero Desktop Client &lt;VERSIONNUMBER&gt;\</td>
<td>• hosts.xml</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• config.xml</td>
</tr>
<tr>
<td></td>
<td>./local/ share/Four Js/ Genero Desktop Client/&lt;VERSIONNUMBER&gt;/</td>
<td>• webcomponent default sub-directory</td>
</tr>
<tr>
<td>Linux®</td>
<td>~/Library/ Application Support/Four Js/ Genero Desktop Client/&lt;VERSIONNUMBER&gt;/</td>
<td>• dictionaries sub-directory</td>
</tr>
<tr>
<td>Mac®</td>
<td>~/Library/ Application Support/Four Js/ Genero Desktop Client/&lt;VERSIONNUMBER&gt;/</td>
<td>or ~/Library/ Application Support/Four Js/ Genero Desktop Client/&lt;VERSIONNUMBER&gt;/</td>
</tr>
<tr>
<td>Directory name</td>
<td>Directory location</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>AppCacheDir</td>
<td></td>
<td>Contains:</td>
</tr>
<tr>
<td>Windows®</td>
<td>C:\Users&lt;USERNAME&gt;\AppData\Local\Four Js\Genero Desktop Client\cache\</td>
<td>application cache</td>
</tr>
<tr>
<td>Linux®</td>
<td>~/cache/Four Js/Genero Desktop Client/</td>
<td>images</td>
</tr>
<tr>
<td>Mac®</td>
<td>~/Library/Caches/Four Js/Genero Desktop Client/ or /Library/Caches/Four Js/Genero Desktop Client/</td>
<td>ftcache sub-directory</td>
</tr>
</tbody>
</table>

Microsoft® Windows® Security Blocking

The Windows® firewall, enabled by default, detects and blocks the Genero Studio Server and the Genero Desktop Client.

From the network point of view, Genero Studio Server and the Genero Desktop Client (used by Genero Studio to display forms) are both considered servers. When you start Genero Desktop Client, or attempt to connect to Genero Studio Server, the firewall detects this and blocks the programs.

A dialog allows you to unblock the program. Follow the instructions provided by the dialog.

If you select Keep Blocking or Ask Me Later, the firewall continues to block the program(s).

If Keep Blocking has been pressed by mistake, you can change this parameter in the Windows® Firewall settings. Ensure you add Genero Studio, Genero Studio Server and Genero Desktop Client to the list of exceptions or allowed apps, and make sure that their listings have a check mark.

Applications

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

- The Shortcut System on page 19
- Customize your Login Box on page 29
- Connections Panel on page 31
- The Terminals Panel on page 33
- The Debug Panel and the logging system on page 35

The Shortcut System

The Genero Desktop Client (GDC) can store the information needed to start an application. The information is stored as a shortcut.

Add a shortcut for each application you want the user to launch.
The GDC monitor must be in administration (admin) mode to create or modify shortcuts. To start the GDC monitor in admin mode, use the --admin or -a command line option.

By default, the Genero Desktop Client starts in user mode, where shortcuts and options cannot be modified. Shortcuts are stored the same way internally across platforms.

![Shortcut Panel]

**Figure 10: Shortcut Panel**

**Creating Shortcuts using the Shortcut Wizard**

To assist you when setting up a new shortcut, a Shortcut Wizard is provided.

When creating a shortcut with the Shortcut Wizard, in step 1, you must choose the type of shortcut connection type that you want to use from the following 3 options:

**Table 3: Shortcut connection types**

<table>
<thead>
<tr>
<th>Shortcut connection type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct, connection is established through terminal emulation</td>
<td>With a Direct connection, the Runtime System is directly connected to the GDC using TCP/IP network. The Runtime System is on a distant host. The GDC will start it via telnet or SSH.</td>
</tr>
<tr>
<td>HTTP, through a web server</td>
<td>With an HTTP, through a web server connection, the GDC connects to the Runtime System via the Genero Application Server using the HTTP protocol. The Runtime System is on a distant host. The GDC will connect to it via Genero Application Server.</td>
</tr>
<tr>
<td>Local execution</td>
<td>With a Local execution shortcut, the Runtime System is on the same computer as the GDC. The Runtime System is on the same host. The GDC will start it as a local application.</td>
</tr>
</tbody>
</table>
Consider these options before you open the Shortcut Wizard. When you have decided on a Shortcut Connection, choose the corresponding task below and follow the instructions to assist you in completing the steps in the Shortcut wizard.

- Create a Direct Connection shortcut on page 21
- Create a HTTP Connection shortcut on page 25
- Create a Local Execution shortcut on page 27

Create a Direct Connection shortcut
This procedure guides you through the process of creating a Direct Connection Shortcut using the Shortcut Wizard.

To open the Shortcut Wizard, in the Shortcuts window, click the New... button.

A direct connection is a connection that is established through terminal emulation.

Shortcut Wizard page 1: Shortcut identification and Connection type
1. Complete the fields of the Shortcut identification section:
   a) In the Name field, provide a name for the shortcut.
   b) Optional: In the Icon field, provide a file name that will be used to display an icon associated with this shortcut.
   c) Optional: If you want to store the shortcut locally for the current user, select the Store shortcut in settings checkbox.

   By default, shortcuts are saved in the &AppDataDir/config.xml file (see GDC configuration file directories on page 18 for more information). Shortcuts written to this file are shared amongst all users who use this installation of the GDC. Selecting the Store shortcut in settings option is useful when the GDC is on a shared network drive and you do not want the current user to modify the common shortcuts.

   Note: When the config.xml file is read-only, the Store shortcut in settings checkbox is selected by default and you do not have the option to deselect it.

   Note: Any modification to a non-local shortcut displays a warning and creates a local copy of the shortcut.

2. In the Connection type section, select the Direct, connection is established through terminal emulation and click Next.

Shortcut Wizard page 2: Host information
3. In the Name field, enter the hostname where the Runtime System is hosted. This can be omitted if you use the -Host or -H command line option.

4. In the Command field, enter the command line that will be executed to start the application on the Runtime System side and click Next.

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

Table 4: Tags for use at the command line

<table>
<thead>
<tr>
<th>Tag</th>
<th>Replaced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>@FGL</td>
<td>FGLSERVER=&lt;IP Address&gt;:&lt;serv num&gt; export FGLSERVER; FGLGUI=1; export FGLGUI</td>
</tr>
</tbody>
</table>

   Note: Additional environment variables are set to assist with the security key mechanism. See Security levels on page 98.

The command to start the demo application using the @FGL tag would be "@FGL; fglrun demo".
Table 5: You can use one of the @FGL variants depending on your system

<table>
<thead>
<tr>
<th>Tag</th>
<th>Replaced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>@FGLNT</td>
<td>set FGLSERVER=&lt;IP Address&gt;:&lt;serv num&gt;&amp;&amp;set FGLGUI=1&lt;br&gt;<strong>Note:</strong> Additional environment variables are set to assist with the security key mechanism. See Security levels on page 98.</td>
</tr>
<tr>
<td>@FGLCSH</td>
<td>setenv FGLSERVER &quot;&lt;IP Address&gt;:&lt;serv num&gt;&quot;;setenv FGLGUI 1&lt;br&gt;<strong>Note:</strong> Additional environment variables are set to assist with the security key mechanism. See Security levels on page 98.</td>
</tr>
<tr>
<td>@FGLKSH</td>
<td>FGLSERVER=&quot;&lt;IP Address&gt;:&lt;serv num&gt;&quot;;export FGLSERVER;FGLGUI=1;export FGLGUI&lt;br&gt;<strong>Note:</strong> Additional environment variables are set to assist with the security key mechanism. See Security levels on page 98.</td>
</tr>
<tr>
<td>@SRVNUM</td>
<td>&lt;GDC listening port - 6400 (The second part of FGLSERVER)&gt;</td>
</tr>
<tr>
<td>@PORT</td>
<td>&lt;GDC listening port&gt;</td>
</tr>
<tr>
<td>@USR</td>
<td>&lt;Client current user name&gt;&lt;br&gt;<strong>Note:</strong> On Windows® operating systems, @USR uses GetUserUserNameEx with &quot;NameUserPrincipal&quot; as the first argument and without the &quot;@domain&quot; part of the result string. On non-Windows® operating systems, @USR and @LEGACYUSR return the same string.&lt;br&gt;</td>
</tr>
<tr>
<td>@LEGACYUSR</td>
<td>&lt;Client current user name&gt;&lt;br&gt;<strong>Note:</strong> On Windows® operating systems, @LEGACYUSR uses GetUserName. On non-Windows® operating systems, @USR and @LEGACYUSR return the same string.&lt;br&gt;<strong>Tip:</strong> @USR should be used in most cases. @LEGACYUSR exists to handle issues that may arise when migrating to Genero Desktop Client 3.0 from an earlier version.</td>
</tr>
<tr>
<td>@LUSR</td>
<td>&lt;Client current user name, lower case version&gt;</td>
</tr>
<tr>
<td>@USER</td>
<td>&lt;User name on the remote system&gt;</td>
</tr>
<tr>
<td>@IP</td>
<td>&lt;IP address of the client computer&gt;</td>
</tr>
<tr>
<td>@COMPUTER</td>
<td>&lt;Machine host name&gt;</td>
</tr>
<tr>
<td>@E_SRV</td>
<td>export FGLSERVER</td>
</tr>
<tr>
<td>@4GLSRVVER</td>
<td>&lt;GDC version&gt;</td>
</tr>
</tbody>
</table>

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

**Shortcut Wizard page 3: Terminal protocol and Terminal options**

5. In the Terminal protocol section, select from the list of options and click Next.

With a direct connection type, the basic mechanism (without any port forwarding configuration) is using 2 connections:

- From GDC to server: the GDC establishes the connection to a server using either the telnet protocol, the SSH protocol or the SSH2 protocol via the fgltty terminal.

  **Tip:** The SSH2 protocol is recommended for security purposes.
• From server to GDC: the server, where the Genero environment is installed, executes a command line which starts the application on the GDC via a TCP/IP network. The IP address of the GDC is retrieved using the FGLSERVER environment variable.

  Note: The telnet, SSH and SSH2 protocols are only used for establishing the first connection from GDC to server.

Using SSH or SSH2, port forwarding can be established to secure your connection. When you use this option, a SSH tunnel is created. This means that, in opposition with the basic mechanism without port forwarding, there are no longer two connections, but a single connection: when the server establishes the connection to the client, it can use the existing SSH connection to tunnel the graphical connection.

  Note: While GDC 3.00 supports IPv6, as DVM does not support IPv6, you cannot launch an application on a distant host with a GDC listening using a direct connection.

6. In the Terminal options section, choose any of the available options and click Next.

   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.

   If Show terminal window 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 Start command in a new shell option allows you to start a regular shell session before executing the remote host command.

   Note: This option is mandatory when using Automatic port forwarding, which can be selected in step 4.

Shortcut Wizard page 4: Port forwarding mode

7. Select the port forwarding method you want from the list and click Next.

   The following port forwarding options are available:
   • If you select Disabled, port forwarding is disabled.
   • If you select Automatic, the option to edit the Port range is provided.
   • If you select Fixed port, you must enter the port that you want to be used.
   • If you select Command line port request, you must enter the command to be executed on the remote host.

   Note: The command must display a string in the form of port=xxx.

   • If you select HTTP port request, you must enter the URL that you want to open for port forwarding port resolution.

   Note: The URL body must include a terminal string in the format of port=xxx.

Shortcut Wizard page 5: Login form and Authentication method

8. In the Login form section, enter a filepath in the Form file field.

   To use your own login form, specify the login form file to use. The form file must be a .ui file, which is a Qt designer's file format. See Customizing your own login form. Check Always on Top to force the login form to always display on top.

9. Choose you Authentication method and click Next.

   The Authentication method will either be Standard or Kerberos.

   • The Standard authentication method:

      In the User field, provide the username you are using to connect to the host. This can be omitted if you use the -User or -U command line option.

      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.
The next two options concern the keeping of the password:

- If **Keep password** is checked, GDC keeps in memory the password you enter the first time you start a shortcut, and reuses the password when you restart. The password is stored for the session; it is kept in memory and is lost if you stop the GDC. The password is kept in memory while GDC is launched and automatically completed in the password field, but it is forgotten once GDC is stopped.

- If **Allow persistent save** is checked, GDC keeps the password between sessions. The GDC can be stopped and re-started, and the password is maintained. This option is only enabled if the **Keep password** option is checked.

  **Important:** GDC never stores your password in a file or elsewhere unless **Allow persistent save** is checked. 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.

The next two options involve the display of the login form:

- If **Don't ask again** is checked, GDC only displays the login box to ask for the password the first time a shortcut is launched. After that, the password will be silently sent without bothering the user with another login box, for the duration of the GDC session.

- If **Even after restart** is checked, the GDC uses the saved password (see Allow persistent save) to silently send the password in subsequent sessions; the password field is no longer displayed, even after the GDC is restarted. This option is only enabled if both **Allow persistent save** and **Don't ask again** are checked.

If any of those options save a password, it will be stored until manually cleared.

The **SSH key file** field: 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 **Kerberos** authentication method:

  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.

**Shortcut Wizard page 6: Terminal strings**

10. Specify the **connection strings** settings and click **Next**.

On this page, the wizard allows you to specify **connection strings**. A table shows a default set of connection string/action items. Defining strings with associated actions configures the GDC with actions to take when the runtime system host displays a given string on the terminal. For a given string the GDC can perform one of 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
- End the terminal

**Tip:** You can select these actions from the drop-down menu in the row under the **Action** column. You can also define new strings and associate them with appropriate actions by clicking **New**.
The default terminal strings should be suitable in most cases, but you may have to adapt them to your system. For instance, the default string to send the command that is last login: may be different on your server.

You can specify whether each string should be recognized only once or every time (check only once).

When the Ignore remaining strings option is allowed and is checked for a string, the rest of the strings that appear in the list below it are ignored. For example, if Ignore remaining strings is checked for password, and the string is parsed from the terminal output, then strings defined below it in the list will not be searched in the terminal output anymore (regardless of any option defined for those strings in the shortcut wizard).

Table 6: Connection string examples

<table>
<thead>
<tr>
<th>Recognized string</th>
<th>Description</th>
<th>Action performed by GDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>password:</td>
<td>This is the string used by the telnet daemon to ask for the password.</td>
<td>Sends the password</td>
</tr>
<tr>
<td>last login:</td>
<td>This is the string used by the telnet daemon to tell the user he has logged in successfully.</td>
<td>Sends the command</td>
</tr>
<tr>
<td>login:</td>
<td>This is the string displayed by the telnet daemon when the login has failed.</td>
<td>Displays a message &quot;Authentication has failed&quot;</td>
</tr>
</tbody>
</table>

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

**Shortcut Wizard page 7: Fgltty Configuration**

11. Configure your Fgltty settings and click Finish to complete the setup and exit the Shortcut Wizard.

Starting with Genero 2.30, these options are inherited from PuTTY. If you need more details on these options, please consult the PuTTY documentation.

**Create a HTTP Connection shortcut**

This procedure guides you through the process of creating a HTTP Connection Shortcut through a web server using the Shortcut Wizard.

With an HTTP connection shortcut, the GDC connects to the Runtime System via the Genero Application Server, using the HTTP protocol.

**Note:** GDC 3.00 supports Internet Protocol version 6 (IPv6) when connected to a Genero Application Server.

An HTTP connection uses a web server.

**Shortcut Wizard page 1: Shortcut identification and Connection type**

1. Complete the fields of the Shortcut identification section:
   a) In the Name field, provide a name for the shortcut.
   b) Optional: In the Icon field, provide a file name that will be used to display an icon associated with this shortcut.
   c) Optional: If you want to store the shortcut locally for the current user, select the Store shortcut in settings checkbox.

By default, shortcuts are saved in the &AppDataDir/config.xml file (see GDC configuration file directories on page 18 for more information). Shortcuts written to this file are shared amongst all users who use this installation of the GDC. Selecting the Store shortcut in settings option is useful when the GDC is on a shared network drive and you do not want the current user to modify the common shortcuts.

**Note:** When the config.xml file is read-only, the Store shortcut in settings checkbox is selected by default and you do not have the option to deselect it.

**Note:** Any modification to a non-local shortcut displays a warning and creates a local copy of the shortcut.

2. In the Connection type section, select the HTTP, through a web server and click Next.
Shortcut Wizard page 2: Web server and HTTP Proxy mode

3. In the URL field, enter the URL for the application that you want to access.
   • When accessing applications using a web server, a typical URL would be: http://myserver/gas/ua/r/gdc-demo.
   • When accessing applications without using a web server, a typical URL would be: http://myserver:6394/ua/r/gdc-demo.

   Note: If the URL of the web server that you enter begins with https (secure), a fourth page of the wizard is automatically added that provides the option to edit the client certificate mode. See Optional: Shortcut Wizard page 4: Client certificate mode.

4. Select a HTTP proxy mode from the list and click Next.

   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.

Shortcut Wizard page 3: Login form and Authentication

5. In the Login form section, enter a file path in the Form file field. Optionally, you can select the Always On Top checkbox.

6. Enter a user in the User field and choose any of the password display options. Optionally, you can enter a realm in the REALM field.

Optional: Shortcut Wizard page 4: Client certificate mode

7. If the URL that you entered in step 3 began with https, you can specify a client certificate mode to authenticate the client to the https server from the following options:
   • Disabled
   • Use certificate/key files

      Note: Currently, 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.
      • 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 password options you selected in Step 1.

      Important: When credentials are required for connecting to an application, the Genero Desktop Client attempts to use single sign-on, in order to avoid requiring the user to enter a password. If single sign-on fails, the Genero Desktop Client switches to the NTLM authentication protocol. The Genero Desktop Client only supports NTLM v1.

      If you are using the NTLM authentication protocol with a Microsoft™ IIS Web server, you must verify that NTLM v1 is also supported. Starting with Microsoft™ IIS Web server 7.0 (Windows® 2008 server), NTLM v2 is required and the Genero Desktop Client is not compatible.

      See the Genero Application Server User Guide for more information on configuring applications.
   • Use system certificates

      Note: 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

8. Click Finish to complete the setup and exit the Shortcut Wizard.
Create a Local Execution shortcut
Create a local execution shortcut using the Shortcut Wizard.

Shortcut Wizard page 1: Shortcut identification and Connection type
1. Complete the fields of the Shortcut identification section:
   a) In the Name field, provide a name for the shortcut.
   b) Optional: In the Icon field, provide a file name that will be used to display an icon associated with this shortcut.
   c) Optional: If you want to store the shortcut locally for the current user, select the Store shortcut in settings checkbox.

   By default, shortcuts are saved in the &AppDataDir/config.xml file (see GDC configuration file directories on page 18 for more information). Shortcuts written to this file are shared amongst all users who use this installation of the GDC. Selecting the Store shortcut in settings option is useful when the GDC is on a shared network drive and you do not want the current user to modify the common shortcuts.

   Note: When the config.xml file is read-only, the Store shortcut in settings checkbox is selected by default and you do not have the option to deselect it.

   Note: Any modification to a non-local shortcut displays a warning and creates a local copy of the shortcut.

2. In the Connection type section, select the Local execution.
3. Click Next.

Shortcut Wizard page 2: Local execution information
4. In the Executable file field, enter the name of the executable to start.

   To start your program on the Runtime System, GDC will simply start an executable (giving it some parameters). This executable will typically be one of the following types:
   • fglrun started in the application directory
   • a batch file that will start all applications

5. In the Working directory field, enter the name of the working directory. This is typically the directory that holds your executable programs.

6. In the Parameters field, enter any needed parameters, such as the name of the program to execute.

   This table shows some possible examples for completing the Local execution information page:

   **Table 7: Local execution shortcut examples**

<table>
<thead>
<tr>
<th>Executable file</th>
<th>Working Directory</th>
<th>Parameters</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>fglrun</td>
<td>/home/fgl-demo/</td>
<td>stores.42m</td>
<td>fglrun should be in the PATH</td>
</tr>
<tr>
<td>c:&lt;mydir&gt;\fgl\bin\fglrun.exe</td>
<td>c:\genero\demo</td>
<td>stores.42m</td>
<td>stores.bat is a batch file that sets the environment and starts the program.</td>
</tr>
<tr>
<td>c:\demos \stores.bat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. Click Finish to create the shortcut and exit the Shortcut Wizard.

The shortcut is created and ready to use.

Important: Environment variables must be set prior to starting the application. Set the variables in the Environment Variables system dialog.
Shortcut Management

When in administration mode, you can edit, duplicate, import, export and remove shortcuts.

Starting shortcuts

Shortcuts can be started from the Shortcuts window in the user interface or from the command line.

In the user interface, there are two options for starting a shortcut:

• Double-click the Shortcut icon in the list.
• Highlight the shortcut in the list by clicking on it once and then click the Start button.

Shortcuts can also be started via the command line:

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

Duplicate shortcuts

To create a copy of a shortcut, select the shortcut and click on the Duplicate... button.

If only one shortcut is selected, a copy is created and the Shortcut Wizard opens, allowing you to modify the new copy.

If several shortcuts are selected, copies are created with unique new names, however the Edit Shortcut wizard is not displayed by default.

Remove shortcuts

To delete a shortcut, select the shortcut and click on the Delete button.

You are asked to confirm the delete. If you answer in the affirmative, the shortcut is removed.

Export shortcuts

Shortcuts can be exported as a .gdc file. A .gdc file is an XML file containing the configuration details for one or more shortcuts. This file can then be used to transfer shortcuts between GDC installations.

To export a shortcut, the GDC monitor must be in admin mode. To start the GDC monitor in admin mode, use the `--admin` or `-a` command line option.

To export a collection of shortcuts:

1. Select one or more shortcuts.
2. Right-click in the shortcuts list and select Export... from the contextual menu.
3. Provide a file name in the Choose a new shortcut file dialog.
4. Click Save. A shortcut file with a .gdc extension is created.

Alternatively, you can export a collection of shortcuts using drag-and-drop:

1. Select one or more shortcuts. Use the SHIFT or CTRL key to select multiple files.
2. Drag the selected shortcuts and drop them into a file browser. A .gdc file is created for each shortcut exported.

When exporting shortcuts using drag-and-drop, if the name of the shortcut contains special characters, the export fails if the operating system does not allow the special characters in the file name as GDC will always attempt to create the file using the same name as the shortcut name. For these shortcuts, use the Export... menu method.

Import shortcuts

Shortcuts can be imported from a .gdc file. A .gdc file is an XML file containing the configuration details for one or more shortcuts. This file can then be used to transfer shortcuts between GDC installations.
To import a shortcut, the GDC monitor must be in admin mode. To start the GDC monitor in admin mode, use the --admin or -a command line option.

When importing, if a shortcut exists with the same name as the importing shortcut, the imported shortcut name will include an incrementing number in parenthesis as part of its name.

To import a collection of shortcuts:

1. Right-click in the shortcuts list and select **Import...** from the contextual menu.
2. Select a shortcut file (.gdc) using the **Choose a shortcut file** dialog.
3. Click on the **Open** button. The imported shortcuts appear in the shortcut list.

Alternatively, you can import a collection of shortcuts using drag-and-drop:

1. Select one or more .gdc shortcut files from the file browser. Use the SHIFT or CTRL key to select multiple files.
2. Drag the file and drop them into the Shortcuts panel of the GDC monitor. The imported shortcuts appear in the shortcut list.

**Shortcuts and environment variables**

You can replace strings with the values of environment variables.

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

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct</td>
<td>host, username, commandline</td>
</tr>
<tr>
<td>http</td>
<td>url</td>
</tr>
<tr>
<td>local</td>
<td>command line, working directory, parameters</td>
</tr>
</tbody>
</table>

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\%).

**Customize your Login Box**

Customize the login box for your own branding or look-and-feel.

Qt Creator provides a default login box.

**Figure 11: Default login box**
Before you begin

Download Qt Creator, available from the Qt website. Download the version for your operating system.

Create a blank form widget

To create a custom login box from Qt Creator, run the integrated Qt Designer. To start:

1. Open Qt Creator.
2. Select File > New File or Project.
3. Under File and Classes select Qt. Select Qt Designer Form. Click Choose....
   The form Choose a Form Template opens.
4. Under templates\forms, select Widget. Click Next....p
   The Location form opens.
5. Enter a Name for your .ui file and the Path where it will be stored. Click Next.
   The Project Management form opens.
6. Click Finish.

Create the form

In Qt Designer Form, create a form with 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

Assign the custom login box

Assign the form you created (the .ui file) to a shortcut.

You assign the form:
- in step 5 when creating a Direct connection shortcut.
- in step 3 when creating an HTTP shortcut.

![Figure 12: Specify the Login form](image)

Once assigned, your form is used instead of the default login box.
**Tips**

- As with a Genero layout, use a Vertical Layout and Horizontal Layout to correctly align and organize your widgets.
- We strongly recommend you embed all elements in a Grid layout (QGridLayout). The Genero Desktop Client (GDC) always resizes the Login Box to its minimum size. When previewing (Alt+Shift+R or **Tools > Form Editor > Preview**) your form in the Qt Designer, you should not be able to resize it to a very tiny size; using a grid layout around the various items helps to avoid this. Another 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 can add widgets typically not used in a login box, such as TextEdit or RadioButton widgets.

**Sample**

This login box is produced by a customized .ui file. You can request a copy of this file from your local support.

![Figure 13: Customized login box example](image)

**Connections Panel**

The Connections panel lists applications and cookies being handled by the Genero Desktop Client (GDC).

**Overview**

The Connections Panel is comprised of two sections: **Connections** and **Cookies**.
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**
The connection type: Direct, HTTP or local execution.

**Date**
When the application was started.

The **Switch to** button raises the selected application to the top, and the focus sets on the current window.

**Tip:** This feature allows you to find your application easily when many programs are launched.

The **Close** button stops the selected application(s). When clicked, the information is sent to the Runtime System and the application is stopped by the GDC; the **Close All** button closes all running applications.

**Important:** GDC will first send a close request to the runtime system, which may be interpreted differently depending on your Genero application settings; see OPTIONS ON CLOSE APPLICATION in the *Genero Business Development Language User Guide*, and will close the network connection after a given delay if the Runtime System does not react.
Cookies
For each cookie, it displays:

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the cookie.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>The path to the cookie.</td>
</tr>
<tr>
<td>Domain</td>
<td>The domain of the cookie.</td>
</tr>
<tr>
<td>Expiration Date</td>
<td>Date when the cookie expires.</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the cookie.</td>
</tr>
</tbody>
</table>

The Clear button clears the selected cookie. The Clear All button clears all cookies.

The Terminals Panel
The terminal utility's main purpose is to launch programs with the parameters configured in shortcuts.

Topics:

- Overview
- Show/Hide
- Close / Close All

Overview
Shortcuts use a terminal emulation utility (called fgltty) 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.
The terminal utility provided is called fgltty.

**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 the logging system

The debug facility for the Genero Desktop client includes logging and the debug console.

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

![Debug Panel](image)

**Figure 16: Debug panel**

**Important:** The **Debug Panel** is only available in **debug mode**.

Using the logging system

Logging can assist with debugging and creating demos.

**Important:** Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

When GDC is started in **debug mode**, logging is available. Logging will help you to:

- Debug your applications
- Create a demo

The complete communication between the front end and Runtime System is logged, so the Runtime System is not needed to replay the 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; user interactions (copy / paste, cursor, and so on) are not saved.
Recording Demo

To record a demo, specify the path and name of a log file to store the scenario. If the file exists, it will be overwritten. Click on the Record button to start the recording. Click on the Stop button to stop the recording.

Important: Only those applications launched AFTER starting the recording are logged.

Replay Demo

To replay a demo, select the log file where the scenario is stored. Click on the Play button to start playing the demo. Click the Pause button to pause the replay. The progress bar indicates the progress of the demo.

Important: No user interaction is possible when replaying a demo. You may have to stop recording the demo before the end of the application. In this situation, use the Connections panel to kill the application.

Viewing the Debug Console

The debug console displays color-coded information about your session.

Figure 18: Console configuration in the Debug panel

The Console section of the Debug Panel defines the verbosity of the messages to appear in the Debug Console. Select the type of messages to display, and use the sliding bar to set the verbosity level.

The Toggle debug verbosity check box alternates between the least verbose and most verbose settings.

Click on the Show... button and the Debug Console window opens.

Important: The Debug Console is only available in debug mode.
The Debug Console displays debug information, categorized by color:

- **Blue**: What is sent by the GDC to the Runtime System.
- **Black**: What is received by GDC from the Runtime System.
- **Green**: Comments or other information.
- **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. The communication threads are also reported individually in the additional tabs, one for each application.

If you want the **Debug Console** to stay in foreground and have it always visible, check **Stays On top**.

### Features

The features of the Genero Desktop Client.

- **The Command Line** on page 38
- **Printing a screen shot** on page 42
- **Local actions** on page 43
- **Localization encoding list** on page 44
- **Accessibility** on page 45
The Command Line

Using the command line with the Genero Desktop Client.

- Command line options on page 38
- Command line examples on page 41

Command line options

The command line options of the Genero Desktop Client, organized by category.

Table 9: Genero Desktop Client command line options: Information

This table presents Genero Desktop Client command line options. Some options share the same attribute and description - these options are interchangeable.

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td></td>
<td>Displays About Box and exits.</td>
</tr>
<tr>
<td>-V</td>
<td></td>
<td>--Version</td>
</tr>
<tr>
<td>-c</td>
<td></td>
<td>Defines an additional configuration file. See Apply an additional configuration file.</td>
</tr>
</tbody>
</table>

Table 10: Genero Desktop Client command line options: Network, System

This table presents Genero Desktop Client command line options. Some options share the same attribute and description - these options are interchangeable.

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n</td>
<td></td>
<td>Starts a new instance of Genero Desktop Client.</td>
</tr>
<tr>
<td>--new</td>
<td></td>
<td>Genero Desktop Client will listen on the first available port starting with new_port.</td>
</tr>
<tr>
<td>-p</td>
<td>new_port</td>
<td>--port</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Important: If an instance is already running, -p has no effect if -n is not specified.</td>
</tr>
<tr>
<td>-q</td>
<td></td>
<td>If the expected port (either 6400, or port specified by --port) is not available, Genero Desktop Client will stop (exit with -1).</td>
</tr>
<tr>
<td>-D</td>
<td></td>
<td>Starts Genero Desktop Client in debug Mode (debug Tree and debug Console are active)</td>
</tr>
<tr>
<td>-A</td>
<td>security_level</td>
<td>Sets Genero Desktop Client's security level regarding the Runtime System's connection.</td>
</tr>
<tr>
<td>Option</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>--listen</td>
<td>ANY</td>
<td>Specify the network listening mode of the Genero Desktop Client. ANY: Listen to any network for a new connection (old behavior)</td>
</tr>
<tr>
<td></td>
<td>LOCALHOST</td>
<td>LOCALHOST: Listen to localhost only, DVM must be on the same host as Genero Desktop Client or must be on a host connected with port forwarding</td>
</tr>
<tr>
<td></td>
<td>NONE</td>
<td>NONE: No listening at all. Only HTTP connection will work, this is the most secure operating mode.</td>
</tr>
<tr>
<td></td>
<td>AUTO</td>
<td>AUTO: Like LOCALHOST, but Genero Desktop Client will switch back to ANY when a regular direct shortcut (without port forwarding) is used, to allow a connection from outside. The Genero Desktop Client will switch back to LOCALHOST when no more connections and no more terminals are active, after a two minutes timeout.</td>
</tr>
</tbody>
</table>

**Important:** AUTO is the default, which means connections from the outside that are launched without using the shortcut system will not work anymore.

### Table 11: Genero Desktop Client command line options: Start Application

This table presents Genero Desktop Client command line options. Some options share the same attribute and description - these options are interchangeable.

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.gdc file</td>
<td></td>
<td>Starts Genero Desktop Client with the shortcut specified in the .gdc file. If the file contains several shortcuts, it starts with the first alphabetically. See <a href="#">The Shortcut System</a>.</td>
</tr>
<tr>
<td>-S</td>
<td>shortcut_name</td>
<td>If Genero Desktop Client has not been launched, Genero Desktop Client will start minimized; then, the shortcut named shortcut_name will be started.</td>
</tr>
<tr>
<td>-s</td>
<td></td>
<td>If Genero Desktop Client has not been launched, Genero Desktop Client will start minimized, using the information given by -U, -H, -T, -P and -C to connect to a DVM.</td>
</tr>
<tr>
<td>--Start</td>
<td></td>
<td>The specified user name will be used when a Direct Connection starts. This option can be used if you share Genero Desktop Client; then each user can create a link to the bin and differentiate the shortcut that will be launched.</td>
</tr>
<tr>
<td>-U</td>
<td>username</td>
<td>The specified host name will be used when a Direct Connection starts with a defined shortcut (with -S), or starts directly (with -s).</td>
</tr>
<tr>
<td>--User</td>
<td></td>
<td>The specified password will be used when a Direct Connection starts with a defined shortcut (with -S), or starts directly (with -s).</td>
</tr>
<tr>
<td>-P</td>
<td>password</td>
<td>The password specified with -P option will be kept in memory and no longer requested.</td>
</tr>
<tr>
<td>--Password</td>
<td></td>
<td>The specified command_line will be used when a Direct Connection starts with a defined shortcut (with -S) or starts directly (with -s).</td>
</tr>
</tbody>
</table>
### Table 12: Genero Desktop Client command line options: Start Genero Desktop Client

This table presents Genero Desktop Client command line options. Some options share the same attribute and description - these options are interchangeable.

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>--admin</td>
<td>Starts the GDC in admin mode.</td>
</tr>
<tr>
<td>-c</td>
<td>--config</td>
<td>Defines an additional configuration file. See Apply an additional configuration file.</td>
</tr>
<tr>
<td>-M</td>
<td></td>
<td>Starts the GDC minimized.</td>
</tr>
<tr>
<td>-i</td>
<td></td>
<td>Starts the GDC with ignore Stored Settings on.</td>
</tr>
<tr>
<td>-X</td>
<td></td>
<td>Closes the GDC if there is no longer an application or terminal running.</td>
</tr>
</tbody>
</table>

### Table 13: Genero Desktop Client command line options: Logging Mechanism

This table presents Genero Desktop Client command line options. Some options share the same attribute and description - these options are interchangeable.

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-T</td>
<td>--Type</td>
<td>Defines which protocol should be used when an application starts with -s. Values can be: TELNET, SSH, SSH2. Default is SSH2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Default is now SSH2.</td>
</tr>
<tr>
<td>-w</td>
<td>ShowTerminal</td>
<td>Defines whether the terminal window is visible (when --startDirect option is used). The terminal is hidden by default.</td>
</tr>
<tr>
<td>-f</td>
<td>ShowFirstLogin</td>
<td>If a password is provided with --Password, Genero Desktop Client 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.</td>
</tr>
<tr>
<td>-k</td>
<td>--PuttyKey</td>
<td>Allows the user to save the password in a persistent way (It will not be asked again, even if Genero Desktop Client is stopped and restarted).</td>
</tr>
<tr>
<td></td>
<td>--PuttyKey (.ppk)</td>
<td>Uses the given Putty Key File as authentication method when Direct Connection.</td>
</tr>
<tr>
<td>-u</td>
<td>--url</td>
<td>Starts the HTTP Genero application given by the URL.</td>
</tr>
<tr>
<td>-g</td>
<td>--gdcfile</td>
<td>Starts directly the remote .gdc file.</td>
</tr>
</tbody>
</table>
Important: Sensitive and personal data may be written to the output. Make sure that the log output is written to files that can only be read by application administrators.

<table>
<thead>
<tr>
<th>Option</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l</td>
<td>Log file</td>
<td>Starts Genero Desktop Client and replays the given Log File</td>
</tr>
<tr>
<td>--logplay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-L</td>
<td>Log file</td>
<td>Starts Genero Desktop Client and replays all the Log Files inside a given directory</td>
</tr>
<tr>
<td>--logdir</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-r</td>
<td>Log file</td>
<td>Starts Genero Desktop Client, records a log, and saves the given Log File.</td>
</tr>
<tr>
<td>--logrec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-t</td>
<td>delay</td>
<td>By default, replays Log Files at their recording speed. You can change the delay (in milliseconds) between the steps. Important: A delay that is too small will overcharge Genero Desktop Client. Please consider 100 milliseconds as the smallest acceptable value.</td>
</tr>
<tr>
<td>--logtimeout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-W</td>
<td>[address:]Port--webengine-remote-debugging</td>
<td>Defines the TCP port where a Chrome browser can connect to, in order to use the QT WebEngine Developer Tools. Usage examples:</td>
</tr>
<tr>
<td></td>
<td>Number</td>
<td>-W 9000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--webengine-remote-debugging=9000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--webengine-remote-debugging=127.0.0.1:9000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--webengine-remote-debugging 9002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--webengine-remote-debugging 9000</td>
</tr>
</tbody>
</table>

For more information, see Debug Web content on page 48.

macOS™ users

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

- Start the terminal application (Applications, Utilities) then enter: ./[Applications/gdc.app/Contents/MacOS/gdc command_line]. macOS™ 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, Genero Desktop Client will first check whether another instance is already running. If you really want your Genero Desktop Client instance to stop if the port is not available, use –n and –q together. Using –q alone will stop Genero Desktop Client if the port is not free and not being used by another Genero Desktop Client.

Command line examples

Examples of the Genero Desktop Client command line.

- 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 ; fglrun 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 ; fglrun demo and closes the GDC when all the applications or terminals are over.

### Inspecting the AUI tree

You can view the Abstract User Interface (AUI) tree as it built on the front end for use by the Genero Desktop Client (GDC).

Before you begin:

- In order to view the AUI tree, the GDC must be in debug mode.

You can inspect the AUI tree for a running application. Inspecting the AUI tree can assist in the debugging and testing of the application.

1. If the Genero Desktop Client is not running in debug mode, start the GDC in debug mode using the -D option.
2. With the cursor over the current window, simultaneously hold down the Control (CTRL) key and right-click the mouse.
   The Debug Tree window opens.
3. Click on the various nodes in the AUI tree.
   When you click on a node, the attributes and values of those attributes display in the right-hand side of the Debug Tree window. If the selected node corresponds to a visible component of the current window, the element briefly flashes.
4. Click the Close icon in the window title bar to close the Debug Tree window.

### Printing a screen shot

The Genero Desktop Client provides a feature to send the current window to any installed printer.

You can print a screenshot directly from the Genero Desktop Client. No additional tool is required.

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

**Local actions**

GDC defines an additional set of action objects, to complement the regular action options created by the Runtime System.

**Important:** This feature is deprecated, and may be removed in a future version. It is recommended that you avoid binding action views with a local action, and that you avoid changing the action defaults attributes (such as accelerators) for these actions. A motivation for depreciating local actions is to ensure application consistency across the Genero front-ends, and local actions were only available on the GDC. They remain supported by the GDC front-end for backward compatibility.

Regular action objects are created by the Runtime System. GDC defines an additional set of action objects called *local actions*, such as `editcopy`, `editcut` or `editpaste`, or list navigation local actions like `firstrow`, `nextrow`.

Like regular actions, it is possible to customize the local actions with an accelerator, images, comments, and so on.

For a full list of local actions, see the topic *List of local actions (GDC only)* in the *Genero Business Development Language User Guide*. 
Implement a local action

In form files, create action views for local actions:

```markdown
BUTTON btn1: editcopy;
```

When this button is pressed, the currently selected text is copied into the clipboard.

Configure a local action

Local actions can be configured with action defaults (.4ad file):

```xml
<ActionDefault name="editcopy"
  text="Copy"
  comment="Copy to clipboard"
  accelerator="Control-C"
  image="fa-bell-o"
  contextMenu="yes"
/>
```

Localization encoding list

A list of localization encodings supported by the Genero Desktop Client.

Table 14: Localization encoding list

<table>
<thead>
<tr>
<th>Encoding List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Roman</td>
</tr>
<tr>
<td>Big5</td>
</tr>
<tr>
<td>Big5-HKSCS</td>
</tr>
<tr>
<td>EUC-JP</td>
</tr>
<tr>
<td>EUC-KR</td>
</tr>
<tr>
<td>GB18030-0</td>
</tr>
<tr>
<td>IBM® 850</td>
</tr>
<tr>
<td>IBM® 866</td>
</tr>
<tr>
<td>IBM® 874</td>
</tr>
<tr>
<td>ISO 2022-JP</td>
</tr>
<tr>
<td>ISO 8859-1 to 10</td>
</tr>
<tr>
<td>ISO 8859-13 to 16</td>
</tr>
<tr>
<td>Iscii-Bng, Dev, Gjr, Knd, Mlm, Ori, Pnj, Tlg, and Tml</td>
</tr>
<tr>
<td>JIS X 0201</td>
</tr>
<tr>
<td>JIS X 0208</td>
</tr>
<tr>
<td>KOI8-R</td>
</tr>
<tr>
<td>KOI8-U</td>
</tr>
<tr>
<td>MuleLao-1</td>
</tr>
<tr>
<td>ROMAN8</td>
</tr>
</tbody>
</table>
### Encoding List

<table>
<thead>
<tr>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift-JIS</td>
</tr>
<tr>
<td>TIS-620</td>
</tr>
<tr>
<td>TSCII</td>
</tr>
<tr>
<td>UTF-8</td>
</tr>
<tr>
<td>UTF-16</td>
</tr>
<tr>
<td>UTF-16BE</td>
</tr>
<tr>
<td>UTF-16LE</td>
</tr>
<tr>
<td>UTF-32</td>
</tr>
<tr>
<td>UTF-32BE</td>
</tr>
<tr>
<td>UTF-32LE</td>
</tr>
<tr>
<td>Windows-1250 to 1258</td>
</tr>
<tr>
<td>WINSAMII2</td>
</tr>
</tbody>
</table>

---

### Accessibility

There are accessibility limitations with the Genero Desktop Client, when compared to a standard Microsoft™ Windows™ application.

By default, labels and other widgets that cannot receive the focus cannot be read or spoken by the narrator. The end user must force the reading of the entire window. For the default Windows Narrator, this is accomplished by the hot key combination CTRL+SHIFT+SPACEBAR.

With topmenus and toolbars, the narrator does not read item by item, even when the entire reading of the window is selected. The Genero Desktop Client also does not know when a topmenu or toolbar item is hovered or highlighted. As a result, it is recommended to not use topmenus or toolbars where accessibility is important; you should limit the use of the accessibility to action panels/menus and buttons.

---

### Auto Update

You can automate updates to an existing Genero Desktop Client (GDC) installation.

Along with the GDC installer files, a zip archive (available on the Products download page of the Four Js Web site) is provided for each operating system.

**Warning:** Auto Update allows the update application to perform operations on your system. Please review the security steps and recommendations listed in the Security section of this document, to ensure your network and systems are properly protected.

**The auto-update application**

A Genero BDL application is needed to start the update.

The application can perform tasks to ensure that update is needed. For example, this API call checks the version of the installed GDC:

```javascript
ui.interface.getFrontEndVersion()
```
The actual update starts with a front call:

```plaintext
CALL ui.interface.frontcall("monitor", "update", ["path_to_update_file"", (, "warning text to display")], result)
```

The `path_to_update_file` specifies the zip archive containing the update material. A pop-up warning window is displayed if applications (other than the update app) are running, which allows the user to cancel the update process. If applications are running and the user accepts the pop-up warning, the update process closes all applications, exits the monitor, and performs the update.

The default pop-up warning message is shown:

**There are still applications running.**

*Are you sure you want to update?*

In the (optional) argument of the front call, you can provide your own text to display in the pop-up window.

**Note:** Specifying a custom text will not make the pop-up display. The pop-up warning is only displayed if the user has applications running when the update front call is called.

The aim of the updater is to have a clean installation after the update, based on the content of the update material (the zip archive). All files not listed in the update material zip archive will be lost. To provide additional files, add them to the standard update material zip archive, and use this customized zip archive in your update process. If the updater detects files previously installed but modified by the user, they will be preserved.

**Important:** The zip archive file must be on the same host as the GDC. The easiest way to place the file on the same host is with a file transfer, such as `FGL_PUTFILE`.

**How the update process works**

The front call displays a dialog box asking the user to close all his applications. If the user cancels the dialog, nothing happens and the update attempt ends. If the user accepts the dialog, the monitor closes all remaining applications and then closes itself. It also stops the GDC from listening on the port, so that new applications cannot connect to it. An update pop-up then displays, showing the progress of the update.

**Note:** Aside from responding to the prompt, this does not require any action on your part.

1. On Windows® platforms, you are prompted by UAC to accept to run the gdc updater binary (fjsupdater).
2. It unzips the zip archive containing the update material into a temporary directory.
3. It checks the customer modified files in the current installed GDC. It then copies and renames the modified files in the same temporary directory. For example:

   ```text
   $GDCDIR/etc/mymodifiedfile.xxx -> $TEMPPATCHDIR/etc/mymodifiedfile.xxx. [Year-Month-Day-Hour-Minute]
   ```
4. It closes the GDC process, then removes the installed GDC.
5. It moves the temporary directory (containing the new GDC) into the same path as the previous installed one.

When the update completes, the GDC will be launched with same arguments as it was started previously. An informational dialog displays with the results of the update process.

**Include custom files in the archive**

An archive can be modified by adding custom files to it. Added custom files are copied along with the core GDC files by the update process.
To construct an archive, use the "-ry" zip command parameter ("-y" keeps symbolic links).

**Warning:** Four Js does not support files added using this process. Files may be overwritten by future updates. Use at your own risk.

### Simple Auto Update Application

This is a simple example. You may wish to investigate alternate (better) ways to get the information necessary for your application, such as:

- The server-side path to the archive containing the file to update.
- The local path where to copy the file with the `FGL_PUTFILE`, for example. In this example, the local path is hard-coded.
- The version of GDC contained in the archive, in order to compare to the current GDC. The comparison could be excluded completely, if you want to force the update.

```plaintext
IMPORT os

MAIN

DEFINE pathToServerSideUpdateArchive STRING
LET pathToServerSideUpdateArchive = checkForUpdate()
IF pathToServerSideUpdateArchive IS NULL THEN
    MENU "AutoUpdate check" ATTRIBUTES(STYLE="dialog",COMMENT="No need for an update, you are running the last version")
    ON ACTION accept
        EXIT MENU
    END MENU
ELSE
    MENU "AutoUpdate check" ATTRIBUTES(STYLE="dialog",COMMENT="There is a new GDC version available, would like to update your installation?")
    ON ACTION accept
        CALL doUpdate(pathToServerSideUpdateArchive)
        EXIT MENU
    ON ACTION cancel
        EXIT MENU
END MENU
END IF
END MAIN

-- checkForUpdate
-- Return the path to the update archive or NULL if there is no update required
FUNCTION checkForUpdate()
    DEFINE currentFrontEndVersion STRING
    DEFINE currentFrontEndOsType STRING
    DEFINE updateArchivePath STRING
    -- Retrieve front end information
    LET currentFrontEndVersion = ui.interface.getFrontEndVersion()
    CALL ui.Interface.frontcall("standard", "feinfo", ["ostype"], [currentFrontEndOsType])
    -- Compute the path to the update archive
    LET updateArchivePath = computeServerSideUpdateArchivePath(currentFrontEndVersion, currentFrontEndOsType)
    RETURN updateArchivePath
END FUNCTION

-- computeServerSideUpdateArchivePath
-- Compute the path to the update archive stored on your server
-- TODO this function need to be implemented accordingly to your wish
FUNCTION computeServerSideUpdateArchivePath(currentFrontEndVersion, currentFrontEndOsType)
```
DEFINE currentFrontEndVersion STRING
DEFINE currentFrontEndOsType STRING
DEFINE updateArchivePath STRING
-- TODO --
RETURN updateArchivePath
END FUNCTION

-- doUpdate
-- Do the update using the provided archive
FUNCTION doUpdate(pathToServerSideUpdateArchive)
DEFINE pathToServerSideUpdateArchive STRING
DEFINE pathToClientSideUpdateArchive STRING
DEFINE res STRING
-- Compute the path to the update archive on the client and prepare the client
LET pathToClientSideUpdateArchive = computeClientSideUpdateArchivePath()
-- Transfer the archive on the client
CALL FGL_PUTFILE(pathToServerSideUpdateArchive, pathToClientSideUpdateArchive)
-- Run the update
CALL ui.Interface.frontCall("monitor", "update", [pathToClientSideUpdateArchive], [res])
END FUNCTION

-- computeClientSideUpdateArchivePath
-- Make sure the working dir for the update is empty on the client side
-- Return the path for the archive on the client side
FUNCTION computeClientSideUpdateArchivePath()
DEFINE osType STRING
DEFINE workingDirPath STRING
DEFINE ret STRING
CALL ui.Interface.frontCall("standard", "feinfo", ["ostype"], [osType])
IF ostype = "WINDOWS" THEN
LET workingDirPath = "c:\fourjs.tmp"
CALL ui.Interface.frontCall("standard", "execute", ["cmd /C rd /S /Q " || workingDirPath, TRUE], [ret])
CALL ui.Interface.frontCall("standard", "execute", ["cmd /C md " || workingDirPath, TRUE], [ret])
RETURN "c:\fourjs.tmp\updateArchive.zip"
ELSE
LET workingDirPath = "/tmp/fourjs.tmp"
CALL ui.Interface.frontCall("standard", "execute", ["rm -rf " || workingDirPath, TRUE], [ret])
CALL ui.Interface.frontCall("standard", "execute", ["mkdir -p " || workingDirPath, TRUE], [ret])
RETURN "/tmp/fourjs.tmp/updateArchive.zip"
END IF
END FUNCTION

Debug Web content

You can access tools to inspect and debug layout and performance issues of any web content, specifically WebComponents.

Debugging a Web Component with the GDC front-end

To enable web component debugging with GDC, define the TCP port where a Chrome browser can connect to, in order to use the QT WebEngine Developer Tools.
With GDC, the web component debug port can be defined with the `--webengine-remote-debugging` command line option, or by setting the `QTWEBENGINE_REMOTE_DEBUGGING` environment variable, before starting the GDC.

**Note:** The command line option takes precedence over the environment variable.

For both the `--webengine-remote-debugging` command line option and the `QTWEBENGINE_REMOTE_DEBUGGING` environment variable, the syntax is:

```
[address:]port
```

**Tip:** The same debugging features of the QT WebEngine Developer Tools are available when using the `--webengine-remote-debugging` command line option, or when using the `QTWEBENGINE_REMOTE_DEBUGGING` environment variable. See QT documentation for more details about debugging features when `QTWEBENGINE_REMOTE_DEBUGGING` is set.

Define a simple `port` number, or a network interface address and port with `address:port`. The `address:port` syntax can be used to control which network interface to export the interface on, to access the developer tools from a remote computer. If no `address` is specified, it defaults to `localhost`.

After starting the Genero program displaying on GDC, the QT WebEngine Developer Tools can be accessed by launching a Chrome browser at the following URL:

- `http://localhost:port` (when the GDC executes on the current host)
- `http://hostname_or_IP:port` (when the GDC executes on a remote machine)

For example, to restrict the access to the debugging tools to the local machine on port 9000, use the value "127.0.0.1:9000". To (theoretically) access from any computer in your network on port 9000, use "0.0.0.0:9000".

**Related information**


---

### Spellchecking in Web Components

The GDC supports spellchecking in Web Components.

The GDC is bundled with dictionaries for English, French, Spanish, German, Italian, Portuguese, and Czech. At this time, you cannot add custom words to the included dictionaries, nor can you add dictionaries.

By default, spellchecking in the webview of a Web Component is enabled. Right-click in the webview to enable or disable spellchecking, and to select which language (or languages) to use.

**Important:** Spellchecking may be explicitly disabled by the JavaScript in the webview. When spellchecking is explicitly disabled in the JavaScript, nothing in the form file or in the GDC context menu will enable it. The web component itself will still display spellchecking options in the context menu, but enabling spellchecking will have no affect; there will be no red underlining and no word replacement proposals.

For more information about spellchecking and Web Components, see the The fglrichtext web component topic in the Genero Business Development Language User Guide.

---

### Upgrading

These topics talk about what steps you need to take to upgrade to the next release of Genero Desktop Client, and allows you to identify which features were added for a specific version.

- **New features of the GDC** on page 50
- **Upgrade Guides for the GDC** on page 88
New features of the GDC

These topics provide a look back at the new features introduced with each release of the Genero Desktop Client.

- GDC 3.10 new features on page 5
- GDC 3.00 new features on page 50
- GDC 2.50 new features on page 51
- GDC 2.40 new features on page 52
- GDC 2.32 new features on page 58
- GDC 2.30 new features on page 59
- GDC 2.22 new features on page 70
- GDC 2.21 new features on page 73
- GDC 2.20 new features on page 79

GDC 3.10 new features

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 3.10.

Corresponding upgrade guide: GDC 3.10 upgrade guide on page 88.

Table 15: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDC uses Qt 5.9. A key benefit of Qt 5.9 is the new Chromium 56 renderer for WebComponents.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>The auto-update feature allows users to update an existing Genero Desktop Client installation.</td>
<td>See Auto Update on page 45.</td>
</tr>
<tr>
<td>Debugging of Web Components can take advantage of the Qt WebEngine module, which makes it easy to inspect and debug layout and performance issues of any Web content from a Chrome browser.</td>
<td>See Debug Web content on page 48.</td>
</tr>
<tr>
<td>GDC provides a mechanism for bypassing certificate errors.</td>
<td>See Bypassing certificate errors on page 136.</td>
</tr>
<tr>
<td>GDC provides an interface for deleting stored passwords.</td>
<td>See Security configuration options on page 15.</td>
</tr>
<tr>
<td>The GDC provides spellchecking support in Web Components.</td>
<td>See Spellchecking in Web Components on page 49.</td>
</tr>
<tr>
<td>Drag-and-drop to export and import shortcuts when the monitor is in admin mode.</td>
<td>See Shortcut Management on page 28.</td>
</tr>
<tr>
<td>Clear the Web cache to support development when using webviews.</td>
<td>See Web cache on page 10.</td>
</tr>
</tbody>
</table>

GDC 3.00 new features

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 3.00.

Table 16: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>After the installation of a new version of the GDC, a popup allows you to import the configuration (shortcuts and options) of a previous version.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>
Overview

GDC 3.00 is compatible with Genero runtime system (DVM) 3.00.

When using an HTTP connection through the GAS:

- GDC 3.00 should use uaproxy (ua) and requires a Genero runtime system (DVM) 3.00.
- GDC 2.50 should use gdcproxy (ja) and requires a Genero runtime system (DVM) 2.50.

GDC 3.00 supports Internet Protocol version 6 (IPv6), in addition to Internet Protocol Version 4 (IPv4), when:

- Using a web server (connected to a GAS).
- Using port forwarding through an ssh tunnel.

However, as the DVM does not support IPv6, you cannot launch an application on a distant host with a GDC listening, using direct connection.

GDC configuration files, supporting files, and cached files are now written to the User directory, providing each user with their own configuration settings (amongst other things) by default.

The Connections panel displays information about cookies.

Chinese translation is available.

HTTP protocol enhancements:

- Support of single sign-on (SSO) mechanism.
- Support of auto logout.

Web component enhancements:

- Support of URL-based web components.
- Support of call frontcall.

When copying data, you can select part of the text of a non-editable field.

Fgltty is now based on Putty 0.65

GDC is now based on Qt 5.5

Reference

See the Install and License your Genero Products manual for more information.

See Port forwarding on page 105 and the Creating Shortcuts using the Shortcut Wizard on page 20 section.

See GDC configuration file directories on page 18.

See Connections Panel on page 31.

See the Genero Application Server User Guide for more information on the HTTP protocol.

No additional reference.

Table 17: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI enhancement: scrollbar added to the display of an array</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>(matrix with dimension).</td>
<td></td>
</tr>
<tr>
<td>You now have the ability to specify a range of ports using automatic</td>
<td>See the Port Forwarding and Firewalls on page</td>
</tr>
<tr>
<td>port forwarding. Command line port request and HTTP port request</td>
<td>105 section.</td>
</tr>
<tr>
<td>methods are deprecated and likely to be removed in a future version.</td>
<td></td>
</tr>
<tr>
<td>Improved display of a one row MATRIX. The blue background has been</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>removed and is replaced by a darker rectangle.</td>
<td></td>
</tr>
</tbody>
</table>

GDC 2.50 new features

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.50.

Table 17: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI enhancement: scrollbar added to the display of an array</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>(matrix with dimension).</td>
<td></td>
</tr>
<tr>
<td>You now have the ability to specify a range of ports using automatic</td>
<td>See the Port Forwarding and Firewalls on page</td>
</tr>
<tr>
<td>port forwarding. Command line port request and HTTP port request</td>
<td>105 section.</td>
</tr>
<tr>
<td>methods are deprecated and likely to be removed in a future version.</td>
<td></td>
</tr>
<tr>
<td>Improved display of a one row MATRIX. The blue background has been</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>removed and is replaced by a darker rectangle.</td>
<td></td>
</tr>
</tbody>
</table>
With the spell checker, you now have the ability to specify an URL for the dictionary path.

WEBCOMPONENT now supports the SIZEPOLICY attribute.

Starting with the Genero Desktop Client 2.50, The Genero Desktop Client ActiveX (GDCAX) is deprecated. It is recommended that you use a Genero web client instead.

---

**GDC 2.40 new features**

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.40.

### Table 18: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Summary Line introduced.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Support for built-in search and fast-seek features introduced.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

### Table 19: Widgets

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>A new Combobox style attribute, <code>completionTimeout</code>, has been added. This style attribute also applies to RadioGroups.</td>
<td>See the <em>ComboBox style attributes</em> section in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>A new ComboBox style attribute, <code>comboboxCompleter</code>, has been added.</td>
<td>See the <em>ComboBox style attributes</em> section in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>Style decorations 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, and which was not decorated in previous versions.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>
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

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

WebComponent now accepts gzip encoding.

WebComponent now accepts cookies, which could be useful for authentication purposes. Cookies are kept in memory during the lifetime of the GDC.

Table 20: Traditional Mode

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global (from CALL <code>ui.interface.LoadToolBar(())</code>) Toolbar is now displayed in Traditional mode. Form toolbars are still not displayed, as it makes no sense in the traditional mode context.</td>
<td>See the <code>ui.Interface.loadToolBar</code> section in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>Traditional windows can be configured to have a status bar. COMMENT, ERROR and MESSAGES will be displayed there instead of their own LINE.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>MDI Container can now be a container for traditional applications (and &quot;modern&quot; at the same time).</td>
<td>See the Window containers (WCI) section in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>Traditional applications can now have a &quot;Pop-Tree&quot; StartMenu to start sub applications.</td>
<td>See the Window style attributes section in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>
### Table 21: Shortcut mechanism

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduced in 2.30, Automatic port forwarding implied to use &quot;start a new shell&quot; configuration option. This is no longer the case. The feature can now be used with or without starting a new shell.</td>
<td>See Port Forwarding and Firewalls on page 105.</td>
</tr>
</tbody>
</table>

### Table 22: Monitor

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td>See The Debug Panel and the logging system on page 35.</td>
</tr>
<tr>
<td>The new --listen command line option and Active X setListeningMode() API function have been added to configure the tcp server.</td>
<td>See GDC 2.40 migration guide on page 90 for more details.</td>
</tr>
</tbody>
</table>

### Table 23: Miscellaneous

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero Desktop Client displays reports faster due to improvements in 2.40.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>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.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>
GDC 2.40 new language features
This section introduces Genero new features in Genero Desktop Client 2.40.

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

Summary Line Support
Support for Summary Line introduced with Genero BDL 2.40.

Figure 21: Summary line
Built-in search and fast seek
Support for built-in search and fast-seek features introduced with Genero BDL 2.40.

Figure 22: Built-in Search

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 onDataChange, onProperty, onFocus, setData, setFocus, Action
Figure 23: Debug console

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

**Status bar support in traditional mode**

Traditional windows can be configured to have a status bar.

COMMENT, ERROR and MESSAGES will be displayed there instead of their own LINE.
Figure 24: Status bar

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

**Report Viewer: improved performance**
Genero Desktop Client displays reports faster due to improvements in 2.40.

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.

GDC 2.32 new features
Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.32.

**Table 24: General**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>User can manage the image size on all screen elements.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Shows GDC version in windows for file association.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>

**Table 25: Widgets**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display text of toolbar items next to the icon</td>
<td>See the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>DateEdit: Better handling of misformatted date</td>
<td>See the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
</tbody>
</table>
Table 26: Miscellaneous

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>RingMenu/actionPanel: showing the beginning of the text when larger than buttons.</td>
<td>See the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
</tbody>
</table>

GDC 2.30 new features

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.30.

Table 27: General features

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drag &amp; Drop support introduced in BDL 2.30.</td>
<td>See the <em>Drag &amp; drop</em> topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>WebComponent support in BDL 2.30 allows you to add any html-based component to your BDL application.</td>
<td>See the <em>WEBCOMPONENT</em> topics in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
</tbody>
</table>

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”).
<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users of Genero Desktop Client 2.30 can now freeze table columns to ensure they remain visible when scrolling.</td>
<td>See the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>Genero Desktop Client 2.30 supports &quot;left&quot;, &quot;right&quot;, &quot;center&quot;, and &quot;auto&quot; alignment for column headers in a table.</td>
<td>See the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>The <strong>INCLUDE</strong> attribute can now be used with the DateEdit calendar to prevent selection of invalid dates.</td>
<td>See the <strong>DATEEDIT</strong> item type topics in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>End users can now add their own words to the Textedit / Spell Checker dictionary.</td>
<td>See the <strong>TextEdit style attributes</strong> topics in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>Window size will be adjusted to fit an opened form with the &quot;resetFormSize&quot; Form style. 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 <strong>OPEN FORM ... DISPLAY FORM</strong>, 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 &quot;resetFormSize&quot; form style attribute to indicate that the window must adapt its size to the newly opened form.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>&lt;Style name=&quot;Form.f1&quot;&gt; &lt;StyleAttribute name=&quot;resetFormSize&quot; value=&quot;1&quot; /&gt; &lt;/Style&gt;</td>
<td></td>
</tr>
<tr>
<td>Support for message styling has been added in 2.30. When using the <strong>MESSAGE</strong> or <strong>ERROR</strong> statement, you can customize how the information is displayed.</td>
<td>See the <strong>Message style attributes</strong> topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>GDC 2.30 introduces the new style attribute for Window <strong>ignoreMinimizeSetting</strong>. The <strong>ignoreMinimizeSetting</strong> style attribute can be used to prevent a minimized window from being reopened in a minimized state.</td>
<td>See the <strong>Window style attributes</strong> topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>The <strong>FORMAT</strong> attribute can now be used with <strong>SpinEdit</strong> widgets to support leading 0 format.</td>
<td>See the <strong>SPINEDIT</strong> item type topics in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>The Delete and Backspace keys now select NULL in a <strong>ComboBox</strong>.</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>
**Table 29: Monitor**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>fgltty now uses qPutty for easier deployment and better support of macOS™ and Linux™.</td>
<td>See the qPutty documentation.</td>
</tr>
<tr>
<td>fgltty now supports automatic port forwarding for SSH connections. Fgltty is now able to detect a free port which can be used by the Port Forwarding mechanism.</td>
<td>See Port forwarding on page 105.</td>
</tr>
<tr>
<td>The GDC About box now has a <strong>Copy To Clipboard</strong> button, which will fill the clipboard with information that is useful when you contact your support center:</td>
<td>See GDC information for better support on page 69.</td>
</tr>
<tr>
<td>The Clipboard will contain:</td>
<td></td>
</tr>
<tr>
<td>• GDC version information.</td>
<td></td>
</tr>
<tr>
<td>• Command line used to start GDC.</td>
<td></td>
</tr>
<tr>
<td>• Operating system information.</td>
<td></td>
</tr>
<tr>
<td>• Copies of config.xml and hosts.xml.</td>
<td></td>
</tr>
<tr>
<td>When contacting your support center, you can copy/paste this information in your email; this should ease and speed support.</td>
<td></td>
</tr>
</tbody>
</table>

**Table 30: Miscellaneous**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Settings for REPORT TO PRINTER (in case of DBPRINT=FGLSERVER)</strong></td>
<td>See REPORT TO PRINTER settings (DBPRINT=FGLSERVER) on page 63.</td>
</tr>
<tr>
<td>Printer and font settings can be overridden with two new 'standard' frontcalls.</td>
<td></td>
</tr>
<tr>
<td>When using <strong>DBPRINT=FGLSERVER</strong> and <strong>REPORT TO PRINTER</strong>, text reports are printed via GDC. Printer and fonts can be configured using the <strong>Option panel</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Animated GIFs support</strong></td>
<td>See Animated GIFs support on page 66.</td>
</tr>
<tr>
<td>Animated GIFs are supported for the IMAGE widget.</td>
<td></td>
</tr>
<tr>
<td>If you load an image which is an animated gif, it will be displayed as animated.</td>
<td></td>
</tr>
<tr>
<td>Animated GIFs are only available for the IMAGE form item. They are not supported where the image appears due to the IMAGE attribute (in buttons, toolbars, topmenus and so on). For performance reasons, animated GIFs are not supported in TABLE containers</td>
<td></td>
</tr>
</tbody>
</table>
WinMAIL: Specific Port for smtp

The server port can now be configured with a frontcall method when using WinMain and 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 dialog with applications that require data in ASCII and applications that require wide char data such as UTF-16.

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("WINDDDE","DDEPoke",
    [prog,"Sheet1","R1C1","value","UNICODE"],[res] );
```

GDC 2.30 new language features

This section describes new features for Genero Desktop Client 2.30

WebComponent support

WebComponent support in BDL 2.30 allows you to add any html-based component to your BDL application.

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”).

REPORT TO PRINTER settings (DBPRINT=FGLSERVER)
Printer and font settings can be overridden with two new 'standard' frontcalls.

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" of "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.

GDC 2.30 new UI features
This section describes widget-related new features for Genero Desktop Client 2.30

Frozen columns for tables
Users of Genero Desktop Client 2.30 can freeze columns to ensure they remain visible when scrolling.

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.

Right click on the header of the column that should be frozen:
**Figure 25: 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:

```xml
<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**

Genero Desktop Client 2.30 supports "left", "right", "center", and "auto" alignment for column headers in a Table.

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.)

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

**Adapt window to new form size**

Window size will be adjusted to fit an opened form with the "resetFormSize" Form style.

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.

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

### Window: Message nodes support styles

Styles can now be applied to MESSAGE and ERROR statements.

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:

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

**Table 31: MESSAGE and ERROR examples**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE &quot;This is a message&quot;</td>
<td><img src="image" alt="This is a message" /> OVR:...</td>
</tr>
<tr>
<td>ERROR &quot;This is an error&quot;</td>
<td><img src="image" alt="This is an error" /> OVR:...</td>
</tr>
<tr>
<td>MESSAGE &quot;This is a message&quot;</td>
<td><img src="image" alt="This is a message" /> OVR:...</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="This is a message" /> OVR:...</td>
</tr>
<tr>
<td>ATTRIBUTE( STYLE=&quot;yellow&quot; )</td>
<td><img src="image" alt="This is a message" /> OVR:...</td>
</tr>
<tr>
<td>ERROR &quot;This is an error&quot;</td>
<td><img src="image" alt="This is an error" /> OVR:...</td>
</tr>
<tr>
<td>ATTRIBUTE( STYLE=&quot;yellow&quot; )</td>
<td><img src="image" alt="This is an error" /> OVR:...</td>
</tr>
</tbody>
</table>
**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.

**Animated GIFs support**

Animated GIFs are supported for the IMAGE widget.

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

Animated GIFs are only available for the IMAGE form item. They are not supported where the image appears due to the IMAGE attribute (in buttons, toolbars, topmenus and so on). For performance reasons, animated GIFs are not supported in TABLE containers.

---

![Animation](loading.gif)

**GDC 2.30 new monitor features**

This section describes the new monitor features for Genero Desktop Client 2.30.

**fgltty**

GDC relies on `fgltty` to manage the connectivity with remote servers running Genero programs.

The `fgltty` tool is a modified version of PuTTY.

**Warning:** Fgltty has the purpose of initiating the connection and acting as a start for a Genero Program. It should NOT be seen as a general purpose terminal emulator. Attempting to use it as a general purpose terminal emulator can result in unexpected errors, such as incorrect display of Multibyte Character Sets.
Figure 26: fgltty configuration window

Most PuTTY options have been included inside the GDC shortcut configuration wizard. For more information regarding the fgltty configuration options, please refer to Configuring PuTTY in the PuTTY User Manual.

The GDC creates a configuration file which is passed to fgltty. Fgltty relies completely on PuTTY.

**Automatic Port Forwarding**

fgltty now supports automatic port forwarding for SSH connections.

Fgltty is now able to detect a free port which can be used by the Port Forwarding mechanism.
Figure 27: Automatic port selected under Port forwarding mode

With classic port forwarding, fgltty connects once, retrieves the port to use, returns it to GDC, and GDC restarts fgltty with the tunnel configuration. With automatic port forwarding, fgltty 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 fgltty when selecting the auto port option.
Figure 28: 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 that copies useful support information to the clipboard.

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.

**GDC 2.30 new misc features**
This section describes miscellaneous new features for Genero Desktop Client 2.30

**WinMAIL: Specific Port for smtp**
The server port can now be configured with a frontcall method when using WinMain and 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.

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

The default port remains 25.

For more information, see the WinMail documentation in the *Genero Business Development Language User Guide*.

**WinDDE: handling of ASCII/Wide char data**
WinDDE can dialog with applications that require data in ASCII and applications that require wide char data such as UTF-16.

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:

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

For more detail, see the WinDDE documentation in the *Genero Business Development Language User Guide*.

**GDC 2.22 new features**
Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.22.

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

**Table 32: Experimental features**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genero Desktop Client now supports rich text editing with integrated toolbox or rich text local actions.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>In previous versions, TextEdits 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.</td>
<td></td>
</tr>
</tbody>
</table>

**GDC 2.22 new UI features**
This section describes widget-related new features for Genero Desktop Client 2.22.

**Rich Text Editing**
Genero Desktop Client supports rich text editing with integrated toolbox or rich text local actions.

In previous versions, TextEdits 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:
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".

```xml
<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`:

```xml
<StyleAttribute name="textFormat" value="html"/>
<StyleAttribute name="showEditToolBox" value="yes"/>
```
The following textedit supports html and richtext editing.
It provides an embedded toolbox
with most common editing actions

This shows rich text editing capabilities of Genero:
bold, italic, underline, bold and underline, bold and italic, italic and bold and underline
left aligned text
centered text
right aligned text

justified (this line needs a long text so you can see that the text is justified, i.e. left and right aligned) text

• this is
• a list
• with
• bullets

1. this is
2. a decimal

Figure 30: 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 33: Local action names, accelerators, and icons

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Accelerator</th>
<th>Icon Name</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>richtextbold</td>
<td>control-b</td>
<td>textbold</td>
<td>B</td>
</tr>
<tr>
<td>richtextitalic</td>
<td>control-i</td>
<td>textitalic</td>
<td>I</td>
</tr>
<tr>
<td>richunderline</td>
<td>control-u</td>
<td>textunder</td>
<td>U</td>
</tr>
<tr>
<td>richtextalignleft</td>
<td>control-l</td>
<td>textleft</td>
<td>L</td>
</tr>
<tr>
<td>richtextaligncenter</td>
<td>control-e</td>
<td>textcenter</td>
<td>C</td>
</tr>
<tr>
<td>richtextalignright</td>
<td>control-r</td>
<td>textright</td>
<td>R</td>
</tr>
<tr>
<td>richtextalignjustify</td>
<td>control-j</td>
<td>textjustify</td>
<td>J</td>
</tr>
<tr>
<td>richtextlistbullet</td>
<td>None</td>
<td>textlistbullet</td>
<td>B</td>
</tr>
<tr>
<td>richtextlistdecimal</td>
<td>None</td>
<td>textlistnumbered</td>
<td>L</td>
</tr>
</tbody>
</table>
## Upgrading

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Accelerator</th>
<th>Icon Name</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>richtextdecreaseindent</td>
<td>None</td>
<td>textindentdecrease</td>
<td></td>
</tr>
<tr>
<td>richtextincreaseindent</td>
<td>None</td>
<td>textindentincrease</td>
<td></td>
</tr>
<tr>
<td>richtextdecreasefontsize</td>
<td>None</td>
<td>textfontsizedown</td>
<td></td>
</tr>
<tr>
<td>richtextincreasefontsize</td>
<td>None</td>
<td>textfontsizeup</td>
<td></td>
</tr>
</tbody>
</table>

Then you can hide the toolbox using the following `styleAttribute`:

```xml
<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.

### GDC 2.21 new features

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.21.

#### Table 34: General

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can now run your GDC 2.11 application with GDC 2.21 and 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.</td>
<td>See the <a href="#">GDC 2.21 migration guide</a> on page 93.</td>
</tr>
<tr>
<td>The internal HTTP stack has been rewritten to support pre-2.20 features and HTTP retries. 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:</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>
### Table 35: Windows

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Genero Desktop Client automatically displays scrollbars around the form when the window is larger than the desktop size.</td>
<td>See the <em>Window style attributes</em> topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> 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.</td>
<td></td>
</tr>
<tr>
<td>We strongly recommend that you adapt your forms to the smallest desktop size you target; automatic scrollbars should only appear for &quot;accident&quot; cases.</td>
<td></td>
</tr>
<tr>
<td>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:</td>
<td></td>
</tr>
</tbody>
</table>

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

- Navigation 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.

As well as "maximized", windows can be started minimized using `minimized` as the value for the "windowState" style attribute.

See the *windows style attributes* section in the *Genero Business Development Language User Guide*. |
Table 36: Widgets

<table>
<thead>
<tr>
<th>Overview</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateEdit now has a presentation style named &quot;showCurrentMonthOnly&quot;. This style configures whether the calendar shows only the current months, or displays (in light grey) days of the previous and next month.</td>
<td>See the DateEdit style attributes section in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>You can now set a range on the SpinEdit widget with the new attributes valueMin and valueMax.</td>
<td>See the SpinEdit item type section in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>Image field now supports style attribute &quot;alignment&quot; to define where the picture should be located when the container (widget) is bigger.</td>
<td>See the Image style attributes section in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>Window supports new style attribute &quot;toolBarDocking&quot; to define if the toolbars are movable and floatable.</td>
<td>See the windows style attributes section in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

Table 37: Presentation styles

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can now set a point font size with a non-integer value, for example 8.3pt.</td>
<td>See the Font sizes section in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

Table 38: Monitor

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Connection tab now displays both the application name and text.</td>
<td>See the Connection configuration options on page 13 topic.</td>
</tr>
<tr>
<td>Users now receive an error message if the number of user licenses is exceeded. User Limit exceeded is now a default terminal string, so the end user has feedback if the application can’t start because of a license issue.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>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.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Genero Desktop Client 2.21 is now supported on Windows™ 7 platform.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>GDC 2.21 has been adapted to work with WinSSHd 5.0.9 (some changes in the server have been done by Bitvise too, so you'll need to upgrade the server part to at least version 5.0.9.)</td>
<td>No additional reference.</td>
</tr>
</tbody>
</table>
### Table 39: FrontEnd functionCall

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hardcopy&quot; is now available as a frontCall.</td>
<td>See the Hardcopy section in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>&quot;launchurl&quot; frontCall signature has the same signature for a Genero web client and the Genero Desktop Client (GDC)</td>
<td>See the LaunchURL section in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

### Table 40: Miscellaneous

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows® Only: The HardCopy menu item is now available in the system menu for MDI child windows.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>Windows® now uses the MSI installer system.</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>File association and start menu entries help to improve desktop integration:</td>
<td>No additional reference.</td>
</tr>
<tr>
<td>• .gdc files are associated with the Genero Desktop Client to be run directly in your favorite explorer (Windows®, Linux®).</td>
<td></td>
</tr>
<tr>
<td>• Linux® installer creates entries in your desktop start menu.</td>
<td></td>
</tr>
</tbody>
</table>

### Table 41: Experimental features

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is possible to see a Flash application in the pages you display with the Integrated browser.</td>
<td>See Flash support on page 78.</td>
</tr>
<tr>
<td>Compositing allows you to make some fancy effects with window transparency.</td>
<td>See Compositing on page 78.</td>
</tr>
<tr>
<td>Supported added for a Command Link Button using the buttonType button style attribute.</td>
<td>See the Button style attributes topic in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>Two new frontcalls have been added in the standard frontcall library:</td>
<td>See Store and restore current window size on page 79.</td>
</tr>
<tr>
<td>• storeSize asks GDC to store the current size of the current window.</td>
<td></td>
</tr>
<tr>
<td>• restoreSize asks GDC to restore the stored size.</td>
<td></td>
</tr>
</tbody>
</table>

### GDC 2.21 new language features

This section describes new features in the general category for Genero Desktop Client 2.21.

**Internal HTTP stack has been rewritten**

The internal HTTP stack has been rewritten to support pre-2.20 features and HTTP retries.

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 31: HTTP Retries section of Connection tab](image)

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.

**GDC 2.21 new UI features**
This section describes new windows features for Genero Desktop Client 2.21.

**Automatic scrollbars**
The Genero Desktop Client automatically displays scrollbars around the form when the window is larger than the desktop size.

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:

```xml
<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.

**GDC 2.21 experimental features**

This section introduces experimental new features for Genero Desktop Client 2.21.

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**

You can now watch your favorite YouTube video in your 4GL application.

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.

**Compositing**

Compositing allows you to implement partial window transparency.

Windows® Vista and Windows® 7 introduced Compositing.

If you want a semi-transparent window in GDC, use the "blurBackground" style attribute:

```xml
<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.

**Command Link Button**

A Command Link Button refers to a push button with a design that shows the comment directly. When the mouse goes over the button, a gradient effect occurs.

**Note:** The Command Link button was introduced with Windows® Vista.

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

```xml
<Style name="Button.commandLink">
  <StyleAttribute name="buttonType" value="commandLink"/>
</Style>
```
You can now add a Command Link Button style in your application, and use the mouseover effect.

This will work on systems that support the Command Link Button. On older systems, it displays a simple button with the text and the comment, but without the mouseover effect.

**Store and restore current window size**

Two new frontcalls have been added in the standard frontcall library: storeSize and restoreSize.

- `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.

```plaintext
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.

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.

**GDC 2.20 new features**

Discover the new features and changes in functionality introduced with Genero Desktop Client (GDC) 2.20.

**Table 42: General**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qt4 is now the internal library used for the Genero Desktop Client.</td>
<td>See GDC 2.20 migration guide on page 96</td>
</tr>
<tr>
<td>SVG image format is now supported in two ways:</td>
<td>See the Providing the image resource topic in the Genero Business Development Language User Guide.</td>
</tr>
<tr>
<td>• 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.</td>
<td></td>
</tr>
<tr>
<td>• On other items, as they can't be resized, the SVG image is used as a pixmap, as well as all other image formats.</td>
<td></td>
</tr>
</tbody>
</table>
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' tab. See Advanced configuration options on page 9.

**Table 43: Monitor/Shortcut mechanism**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>A customized login box can be created for shortcuts.</td>
<td>See Creating Shortcuts using the Shortcuts Wizard</td>
</tr>
<tr>
<td>A command line option, &quot;-i&quot; or &quot;--ignoreSettings&quot;, has been added to force ignore settings.</td>
<td>See Command line options</td>
</tr>
<tr>
<td>A read only stored settings option has been added on the Advanced Tab. Settings are applied when loading a form, but they are not modified when closing the form.</td>
<td>See Advanced tab</td>
</tr>
<tr>
<td>A command line option, &quot;-r [filename.log]&quot; or &quot;--logrec [filename.log]&quot;, has been added for recording a log.</td>
<td>See Command line options</td>
</tr>
<tr>
<td>SSH2 is default protocol type when -T is not set.</td>
<td>See Command line options and The Shortcut System</td>
</tr>
<tr>
<td>Shortcuts can be exported as and imported from a Genero Desktop Client (gdc) file. GDC can also be started with a .gdc file and the shortcut starts directly.</td>
<td>See The Shortcut System on page 19.</td>
</tr>
<tr>
<td>The security level now ensures the application started by the Genero Desktop Client is the one you started.</td>
<td>See Security levels on page 98.</td>
</tr>
<tr>
<td>Several Connections and Terminals can be selected and closed (instead of closing them one by one).</td>
<td>See Connections Panel on page 31.</td>
</tr>
<tr>
<td>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.</td>
<td>See Connections Panel on page 31.</td>
</tr>
</tbody>
</table>

**Table 44: FrontEnd functionCall**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>The standard frontCall getwindowid has been added.</td>
<td>See the getWindowId topic in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>
| Four new parameters have been added for the functionCall Standard feinfo:  
  • osversion  
  • numscreens  
  • screenresolution  
  • fepath | See the feInfo topic in the Genero Business Development Language User Guide. |
### Table 45: Widgets

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can add a TreeView widget in your Genero application, based on a simple DISPLAY ARRAY.</td>
<td>See the Tree views topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>HTML hyperlinks are now managed in Labels by setting the styleAttribute <code>textFormat</code> to &quot;html&quot;.</td>
<td>See the TextEdit style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>TextEdit can have an integrated search facility with style <code>integratedSearch</code>.</td>
<td>See the TextEdit style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>A spelling checker is included for TextEdits.</td>
<td>See TextEdit: Spell Checker on page 83; also see the TextEdit style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>Folder nodes support a <code>position</code> style attribute.</td>
<td>See the Message style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>Window style attributes <code>actionPanelDecoration</code> and <code>ringMenuDecoration</code> have been introduced to define the decoration for the ActionFrame.</td>
<td>See the Window style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>Menu &quot;popup&quot; supports a &quot;position&quot; style attribute.</td>
<td>See the Menu style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>You can now use the mouse wheel to change the focus of the menu.</td>
<td>No reference.</td>
</tr>
<tr>
<td>The Button node now supports a &quot;buttonType&quot; style attribute.</td>
<td>See the Button style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
<tr>
<td>MDI Containers can now be displayed as &quot;Tabbed MDI&quot; (like FireFox / IE7) with two new style attributes: <code>tabbedContainer</code> and <code>tabbedContainerCloseMethod</code>.</td>
<td>See the Window style attributes topic in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
</tbody>
</table>

### Table 46: Tables

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiSelection in DISPLAY ARRAY enables you to select different lines using the usual key combination (SHIFT, CTRL, and so on)</td>
<td>See the DISPLAY ARRAY topics in the <em>Genero Business Development Language User Guide</em>.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See the Editable record list (INPUT ARRAY) section in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

The TABLE widget can be used to render a Picture Flow, a widget to display images with an animated transition effect.

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See the Table style attributes topic in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See Tables: resizeFillsEmptySpace styleAttribute on page 84; see also the Table style attributes topic in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

### Table 47: Action mechanisms

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>To adapt to Multiple Dialogs and ON ACTION ... IN FIELD, the LocalAction mechanism has been slightly revised.</td>
<td>See the ON ACTION block topic in the Genero Business Development Language User Guide.</td>
</tr>
</tbody>
</table>

The Genero Desktop Client creates local actions prefixed by the screen record, for instance `sr1.nextrow`. This allows you to create ActionViews bound to a specific screen record, such as a dedicated navigation panel for a specific table.

<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>See Qualified Local Actions on page 83.</td>
</tr>
</tbody>
</table>

### Table 48: Experimental features

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Look and Feel</strong>&lt;br&gt;The lookAndFeel style attribute allows you to customize your application.</td>
<td>See Look and Feel on page 85.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Form Layout</strong>&lt;br&gt;The Form Layout is a style applied to a GRID that lets the Genero Desktop Client display the content of the section, ignoring the per alignment and so on. Simple fields are bound to their labels using the TITLE attribute.</td>
<td>See Form Layout on page 86.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overview</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integrated Browser</strong>&lt;br&gt;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 &quot;browser&quot;, your image container becomes a browser. Instead of setting an image name, you can set a URL.</td>
<td>See Integrated Browser on page 87.</td>
</tr>
</tbody>
</table>
GDC 2.20 new language features
This section introduces action mechanisms new features for Genero Desktop Client 2.20.

Qualified Local Actions
The Genero Desktop Client creates qualified local actions.

The Genero Desktop Client creates local actions prefixed by the screen record, for instance `sr1.nextrow`. This allows you to create ActionViews bound to a specific screen record, such as a dedicated navigation panel for a specific table.

GDC 2.20 new UI features
This section describes widget-related new features for Genero Desktop Client 2.20.

TextEdit: Spell Checker
A spell checker can be provided for TextEdit fields in forms displayed in the Genero Desktop Client (GDC).

When used, the GDC underlines unknown words. Right-click on the unknown word to access a list of suggestions for the word.

Providing dictionary files
The spell checker uses Hunspell (see http://hunspell.sourceforge.net/). Hunspell is the default spell checker for OpenOffice.org or Mozilla tools.
You must provide the Genero Desktop Client (GDC) with two dictionary files for each language. These files can be downloaded from http://extensions.openoffice.org/. Extract the files in the oxt archive.

For each language, you create a style for TextEdit fields and specify the two files for the spellCheck StyleAttribute using the following format:

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

You can load dictionary files via a URL, or by using the fgl_putfile() built-in function. If there is an error loading dictionary files, spell checking is simply not done. No error will be reported by the application.

**Note:** Loading the dictionary can take time. For instance, it takes approximately 5 seconds for the French dictionary to load. As a result, spell checking may not be immediate when loading a form.

**Loading dictionary files from a URL**

**Tip:** This is the recommended method for loading dictionary files.

The files can be loaded via a URL. Specify the URLs using the spellCheck StyleAttribute:

```
<Style name="TextEdit.spellfr">
</Style>
```

**Loading dictionaries using fgl_putfile**

The files can be stored on the Runtime System side and can be uploaded to the GDC using fgl_putfile().

By default, files are read from the directory where the GDC executable resides. For example, this style definition specifies the dictionaries for the "US English Spell Checking Dictionary", and the files are expected to be in the directory with the GDC executable:

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

The directoriesDirectory parameter for the standard.feInfo frontcall can be used to get the directory where spell checker dictionary files are to be uploaded. This ensures that the directory is correct for the current version of GDC.

Alternatively, you can specify an absolute path:

```
<Style name="TextEdit.spellUs">
  <StyleAttribute name="spellCheck" value="file:///c:/dicts/en_US.aff|file:///c:/dicts/en_US.dic" />
</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.
This section introduces experimental new features for Genero Desktop Client 2.20. 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 are a skilled C++ developer, you can build your own style, following Qt's guidelines. This gives you an example of what can be done.

A few built-in styles are provided; the list varies based on the platform where the GDC runs. You can view the list of built-in styles provided for a platform by viewing the Information tab in the Genero Desktop Client console's About box.
Figure 34: Viewing the list of look-and-feel presentation styles

Example

This example of a 4st file defines a window style using a "dotnetoffice" look-and-feel.

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

Form Layout

The Form Layout is a style applied to a GRID that lets the Genero Desktop Client 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:

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

The ATTRIBUTES section defines the TITLEs:

```xml
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:

```xml
<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 complex html data, but cannot 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**

![Google maps screenshot](image).

**Figure 35: Google maps screenshot**

Here is the per file:

```perl
LAYOUT (TEXT="Google Maps")
  GRID
  |
  [map ]
  [ ]
  [ ]
  [ ]
  [ ]
  [ ]
```
Here is the corresponding 4st file:

```xml
<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 have 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. We expect the version to be better supported by Qt in the future.

## Upgrade Guides for the GDC

Each upgrade guide is an incremental upgrade guide that covers only topics related to a specific version of Genero. It is important that you read all of the upgrade guides that sit between your existing version and the desired version.

- GDC 3.10 upgrade guide on page 88
- GDC 3.00 upgrade guide on page 89
- GDC 2.5x upgrade guide on page 90
- GDC 2.4x upgrade guide on page 90
- GDC 2.3x upgrade guide on page 91
- GDC 2.2x upgrade guide on page 93

### GDC 3.10 upgrade guide

This section describes differences you may encounter when upgrading to Genero Desktop Client 3.10.

Corresponding new features page: GDC 3.10 new features on page 5.

### Debug WebComponents

The Enable WebComponent debugger check box has been removed from the Debug panel. To debug a WebComponent, you must now set the QTWEBENGINE_REMOTE_DEBUGGING environment variable or the `--webengine-remote-debugging` command line option. See Debug Web content on page 48.

### Remove "User limit exceeded" from existing direct shortcuts

Prior to GDC 3.10.14, the shortcut creation wizard presented the string "User limit exceeded" with an action of "Display a message" when defining terminal strings. This string/action is no longer valid and no longer appears in the wizard for new shortcuts being created.
You should manually edit existing shortcuts (created prior to 3.10.14) to remove this string/action. If the string/action pair is not deleted from existing shortcuts, the risk is that duplicate error message dialogs can display.

**New rendering at the bottom of a TABLE INPUT ARRAY**

Previously, when a table displays during an INPUT ARRAY, non-data rows displayed after the data rows, appearing as a grid at the end of the table. This behavior could cause confusion as there was no visual distinction between a data row that contained no values and a non-data row.

With GDC 3.10, when a table displays during an INPUT ARRAY, only data rows display. If an empty row appears at the end of the table, it is a data row (albeit an empty data row) and is part of the dynamic array. The remainder of the table space displays as a blank area. Clicking on the blank area allows the user append a new row after the last data row.

**Local actions are now deprecated**

Business applications should not have to implement local actions, such as basic copy, cut and paste editor actions. The application should instead focus on business needs, and local actions should simply be part of the front-end functionality. Therefore, local actions are marked as deprecated, starting with Genero 3.10. An additional motivation for deprecating local actions is to ensure application consistency across the Genero front-ends, and local actions were only available on the GDC. See Local actions on page 43.

**GDC 3.00 upgrade guide**

This section describes differences you may encounter when upgrading to Genero Desktop Client 3.00.

**Genero Desktop Client ActiveX (GDCAX) is desupported.**

For new development, use the latest Genero web client.

**New location for configuration files**

Starting with version 3.00, the default location of configuration files has changed. See GDC configuration file directories on page 18. Ensure that any administrative tasks relating to the configuration files take account of the new location.

**Having users share a GDC configuration file**

Starting with version 3.00, the GDC writes its configuration file (config.xml) in the User directory; see GDC configuration file directories on page 18 for the OS-specific path.

If you have been providing your end users with a shared drive installation of the GDC where multiple users (by default) share the same configuration file, you must act. To have users continue to share a common configuration file, use the --config option to specify the shared file. See Apply an additional configuration file on page 18.

**Default images no longer in /pics directory**

Starting with version 3.00, the /pics directory no longer exists in the GDC installation path. If you were using the default images from the /pics directory and you are still using Genero 2.5x, the images will not display. Genero 2.5x does not centralize default images on the runtime side. For further information, see the 3.00 upgrade topics in the Genero Business Development Language User Guide.

**Shift-F10 no longer can be an accelerator**

Shift-F10 is the standard Windows® shortcut to open a contextual menu. With the upgrade to Qt 5.5, it can no longer be used as an action accelerator. If you set an action with acceleratorName="Shift-F10", pressing these keys now opens the contextual menu.
GDC 2.5x upgrade guide
This section describes differences you may encounter when upgrading to GDC 2.5x.

Starting with 2.50, the Genero Desktop Client ActiveX (GDCAX) is deprecated. It is recommended that you use the latest Genero web client instead.

GDC 2.50 migration guide
This topic lists differences you may encounter when upgrading to GDC 2.50.

Genero Desktop Client ActiveX (GDCAX) is deprecated
It is recommended that you migrate any GDCAX applications to use the latest Genero Web Client.

GDC 2.4x upgrade guide
This section describes differences you may encounter when upgrading to GDC 2.4x.

GDC 2.40 migration guide
This topic lists differences you may encounter when upgrading to GDC 2.40.

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 in the Genero Business Development Language User Guide. 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.

GDC 2.3x upgrade guide

This section describes differences you may encounter when upgrading to GDC 2.3x.

GDC 2.30 migration guide

This section describes differences you may encounter when upgrading to GDC 2.30.

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:

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

Now:

```html
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 fgltty 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 fgltty 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 fgltty 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 36: 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).

**GDC 2.2x upgrade guide**

This section describes differences you may encounter when upgrading to GDC 2.2x.

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

**GDC 2.21 migration guide**

This section describes differences you may encounter when upgrading to GDC 2.21.

**Windows™: Installer uses MSI technology.**

The installer and uninstaller have been rewritten using MSI technology. 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, 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:

```plaintext
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 37: GDC behaving like TUI](image-url)
This may have an impact on your programs if you are using actions without names, which was a classic pattern in early Genero Days:

```plaintext
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 38: Extra button with no label](image)

The reason is that

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

is actually:

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

which is:

```plaintext
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:

```bash
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
```

**GDC 2.20 migration guide**

This section describes differences you may encounter when upgrading to GDC 2.20.

**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 theme mechanism was not designed to make use of all the new features introduced by Windows® Vista, macOS™ 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 macOS™ (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

```plaintext
-- 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.

```plaintext
-- 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 font 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 49: RGB values: X11 and W3C comparison

<table>
<thead>
<tr>
<th>name</th>
<th>X11 RGB value</th>
<th>W3C RGB value</th>
</tr>
</thead>
<tbody>
<tr>
<td>green</td>
<td>#00ff00</td>
<td>#008000</td>
</tr>
<tr>
<td>grey</td>
<td>#bebebe</td>
<td>#808080</td>
</tr>
<tr>
<td>maroon</td>
<td>#b03060</td>
<td>#800000</td>
</tr>
<tr>
<td>purple</td>
<td>#00ff00</td>
<td>#a020f0</td>
</tr>
</tbody>
</table>

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.

Security

These topics cover security and the Genero Desktop Client.

• Security levels on page 98
• GDC and SSH on page 102
• GDC and SSH simple setup on page 104
• Port Forwarding and Firewalls on page 105
• Implementing a Secure Server with GDC on page 120
• SSH Configuration Troubleshooting on page 135
• Microsoft firewall configuration on page 138
• Microsoft User Account Control on page 139

Security levels

The security level determines what verification occurs when a connection arrives on a listening port.

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

Change the security level using the `gdc -A` command line or on the Security tab.

Table 50: Security Levels

<table>
<thead>
<tr>
<th>Security Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Security level 0 is the least secure. Command Line: <code>gdc -A 0</code> Any connection started by the runtime system is authorized. No security message displays.</td>
</tr>
<tr>
<td>Security Level</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 1             | Security level 1 displays a warning.  
Command Line: gdc -A 1  
When the runtime system starts a connection:  
1. The GDC checks for the host IP address exists in $AppDataDir/hosts.xml. If the host exists, the GDC accepts the connection and the application displays.  
2. If the IP address is not found, the **Security Connection Message** dialog displays a warning and asks the user whether to connect. |
| 2             | Security level 2 uses a key mechanism.  
**Important:** This only works when using a direct connection shortcut to start an application.  
Command Line: gdc -A 2  
1. The GDC completes the security key mechanism check. If both keys check, then the connection is made.  
2. If the security key mechanism check fails, then the GDC looks for the IP address of the DVM host in $AppDataDir/hosts.xml. If the host exists, then the connection is made.  
3. If this host check fails, then the **Security Connection Message** dialog displays a warning and asks the user whether to connect.  
If the runtime system does not handle this feature, you will not be able to run an application at this security level. |
| 3             | Security level 3 uses a key mechanism.  
Command Line: gdc -A 3  
**Important:** This only works when using a direct connection shortcut to start an application.  
1. The GDC completes the security key mechanism check. If both keys check, then the connection is made.  
2. If the security key mechanism check fails, then the GDC looks for the IP address of the DVM host in $AppDataDir/hosts.xml. If the host exists, then the connection is made.  
3. If this host check fails, then the connection is rejected.  
If the runtime system does not handle this feature, you will not be able to run an application at this security level. |

**The security key mechanism check**

How the key mechanism works:

1. When the GDC starts, it generates two random keys. These are known as UUIDs.
2. When the GDC starts a direct connection and that connection uses one of the @FGL tags (\_FGL, \_FGLNT, \_FGLCSH, or \_FGLKSH) in the command, \_FGLFEID and \_FGLFEID2 are exported. At this point, fglrun has two variables in its environment (\_FGLFEID and \_FGLFEID2) that it will use to verify the GDC it attempts to connect to.  
3. The GDC gets the \_FGLFEID from the DVM via the GUI connection and compares it to the \_FGLFEID it initially generated. If they match, it knows that it is connecting to the correct DVM. Otherwise, it should reject the connection.  
4. The DVM gets the \_FGLFEID2 from the GDC. If it matches the \_FGLFEID2 value set in its environment, it knows it has the correct GDC. Otherwise, it should reject the connection.
When a check fails, the security level ultimately determines whether to reject the connection or to allow the end-user to override and accept the connection.

**The Security Connection Message dialog**

When displayed, the Security Connection Message dialog allows a user to accept a connection despite not passing the security checks involving key mechanism verification or `hosts.xml` validation.

![Security Connection Message dialog](image)

**Warning:** Take caution before allowing your users to bypass the security checks. The key mechanism detects and prevents unauthorized users and applications from connecting via the Genero Desktop Client. Users need to understand the implications and security risks of electing to allow connections that have not been properly validated.

**Table 51: Security Connection Message Options**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Accept this connection and only this connection. The connection information is stored in memory for the duration of the connection. Any additional connection from the same host causes the message to be displayed again.</td>
</tr>
<tr>
<td>Yes to All</td>
<td>The GDC accepts this connection and any other connection from the same host. This setting is saved to <code>$AppDataDir/hosts.xml</code> when the GDC closes.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> You can modify the <code>hosts.xml</code> file if needed, or remove it to clear the authorized list.</td>
</tr>
<tr>
<td>No</td>
<td>The GDC rejects this connection and the application will not run.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> Did you answer &quot;No&quot; by accident? When you next execute the same application, the dialog redisplay.</td>
</tr>
</tbody>
</table>

**Related concepts**

- **Security configuration options** on page 15
  Access the Security tab to configure the security level, clear stored passwords, and view/clear known hosts.

- **GDC configuration file directories** on page 18
  The GDC configuration files are stored in two (default) directories: `AppDataDir` and `AppCacheDir`.

**Related tasks**

- **Create a Direct Connection shortcut** on page 21
  This procedure guides you through the process of creating a Direct Connection Shortcut using the Shortcut Wizard.

**Security terms**

The security section of the documentation uses several terms that must be clarified for a good understanding.

**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

GDC and SSH
This section provides an overview and the prerequisites of using GDC with SSH.
• GDC and SSH overview on page 102
• GDC and SSH prerequisites on page 102

GDC and SSH overview
GDC with SSH provides security and port forwarding.

Figure 40: 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, SSH2 or HTTPS for those connections. Furthermore, any production environment on an intranet or internet should use a secured layer. 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
This topic covers prerequisites and SSH connection options.

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 52: Data encryption selection matrix](#) on page 103 to determine which settings you will need.

**Table 52: Data encryption selection matrix**

<table>
<thead>
<tr>
<th>Type of connection</th>
<th>Command encrypted</th>
<th>GUI encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>telnet</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>ssh</td>
<td>X</td>
<td>NO</td>
</tr>
<tr>
<td>ssh port forwarding</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ssh2</td>
<td>X</td>
<td>NO</td>
</tr>
<tr>
<td>ssh2 port forwarding</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**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 [Table 53: Connection method support matrix](#) on page 103 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.

Table 53: Connection method support matrix on page 103 uses the following 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 53: Connection method support matrix**

<table>
<thead>
<tr>
<th>Firewall or NAT on Server Side</th>
<th>telnet</th>
<th>SSH</th>
<th>SSH + Port Forwarding</th>
<th>SSH2</th>
<th>SSH2 + Port Forwarding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall or NAT on Client Side</td>
<td>2</td>
<td>2</td>
<td>X</td>
<td>2</td>
<td>X</td>
</tr>
</tbody>
</table>
GDC and SSH simple setup

The simple setup assumes that you are on a corporate LAN with no firewalls, and allows for all connection methods.

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

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 fglrun 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

This section covers configuration of Port Forwarding with client or server-side firewalls.

- [Port forwarding](#) on page 105
- [Port forwarding and the client-side firewall](#) on page 111
- [Port forwarding and the server-side firewall](#) on page 114

### Port forwarding

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

**Note:** Genero Desktop Client 3.00 supports Internet Protocol version 6 (IPv6), in addition to Internet Protocol Version 4 (IPv4), when using port forwarding through an ssh tunnel.
Figure 42: Simple connection with Port Forwarding
Figure 42: Simple connection with Port Forwarding on page 106 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 43: Connection to Server side Firewall with Port Forwarding on page 107 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 44: Specify specific port number 2222
In Figure 43: Connection to Server side Firewall with Port Forwarding on page 107 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 45: Specify fixed port number 29000 on page 109 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 environment 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 46: @FGL command example
Port forwarding and the client-side firewall

This section details how to configure port forwarding with a client-side firewall.

Figure 47: 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 47: Connection from client side firewall with port forwarding on page 111). 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 48: Connection from Client side Firewall on page 112). The set up of the router will be covered here; port forwarding is covered in a separate section.
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:

<table>
<thead>
<tr>
<th>Destination Port</th>
<th>Source IP (Internal)</th>
<th>Source IP (External)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6401</td>
<td>192.168.1.10:6400</td>
<td>192.168.1.10:6401</td>
</tr>
<tr>
<td>6402</td>
<td>192.168.1.11:6400</td>
<td>192.168.1.11:6402</td>
</tr>
<tr>
<td>6403</td>
<td>192.168.1.12:6400</td>
<td>192.168.1.12:6403</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Destination Port</th>
<th>Source IP (Internal)</th>
<th>Source IP (External)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6401</td>
<td>192.168.1.10:6401</td>
<td>192.168.1.10:6401</td>
</tr>
<tr>
<td>6402</td>
<td>192.168.1.11:6402</td>
<td>192.168.1.11:6402</td>
</tr>
<tr>
<td>6403</td>
<td>192.168.1.12:6403</td>
<td>192.168.1.12:6403</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Destination Port</th>
<th>Source IP (Internal)</th>
<th>Source IP (External)</th>
</tr>
</thead>
<tbody>
<tr>
<td>213.39.41.73:10000</td>
<td>192.168.0.53:6400</td>
<td>213.39.41.73:10000</td>
</tr>
</tbody>
</table>

The command line in the GDC would look like:

```
FGLSERVER=213.39.41.73:36000; fglrun demo
```
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**

This section details how to configure port forwarding with a 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 50: Connection to server side firewall on page 115 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.

See Figure 51: Connection to Server side firewall with port forwarding on page 116 for an example of how to do this using ssh with port forwarding.
Figure 51: 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 52: Showing configuration for access to Server 1
Figure 53: Showing configuration for access to Server 1
Figure 54: Showing configuration for access to Server 2
Implementing a Secure Server with GDC

Implement a secure server by denying users access to the command line or shell.

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™)

Implementing a Secure Server with GDC

Implement a secure server by denying users access to the command line or shell.

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 on page 121
• Solutions overview on page 121
• The shell script on page 122
• Setup SSH login on page 122
• Setup telnet on page 126
• Password management on page 128
• AUTOPORTFIND source code example on page 130
• Login script on page 134

Prerequisites
This topic discusses prerequisites of configuring a secure server.

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
This topic discusses replacing the login shell to implement a secure server.

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

This topic covers the steps required to replace a login shell with a customized 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

Configure a GDC shortcut to launch the application and implement port forwarding.

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 56: The Genero Desktop Client shortcut entry for using ssh.

In the Command field, we have specified AUTOPORT. This corresponds to an option near the end in the login script.
Figure 57: 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 58: Configuring FJSOORTFORWARD 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 59: Run as user1

When the shortcut is run, it will log in using AUTOPORT first. This will match a case statement in the script, and return a string "FJSPTFORWARD=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

This topic describes the steps to configure GDC connections using telnet.

Telnet doesn't offer port forwarding, so the setup is a bit simpler. But it also doesn't give the flexibility needed when going through firewalls, and offers 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 60: Specify the command line arguments for 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**

Secure server password management.

- Handling expired passwords on page 128
- Changing passwords on page 129

**Handling expired passwords**

This section explains how to configure GDC behavior for 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 62: 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**

Create a shortcut to support password changes.

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:
This section provides an example of the source code to produce the port number for tunnelling with ssh. This script 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 immediately 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=
""\n"Generate a port number for use with port forwarding.\n"\n"-e, --env"
"Send FJS_PORTFORWARD=<port> to stdout."
"\n"-r" Cycle through port assignments to determine which ports\n"the OS assigns to ports when originating connections."
"-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."
"-v" Display the version number."
; 
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
", 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
", 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",
                progsname, *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("FJSORTFORWARD=");
            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
#ifndef UTIL_H
#define UTIL_H
#ifndef 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
#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
#ifdef USE_SOCKETS
#undef USE_SYS_SOCKIO
#define USE_SYS_SOCKIO
#endif
#define SOCKLEN_T int
#endif
#ifdef __hpux
#define SOCKLEN_T int
#endif
}
Login script

This section provides 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.

```bash
#!/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
fi

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

Possible configuration issues when implementing SSH.

- Wireless systems on page 136
- Need to change the port that GDC listens on on page 136
- Sessions expiring on page 136
Wireless systems

Lost signals with wireless connections can cause connection loss.

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

GDC port can be changed when required.

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

Routers may expire sessions.

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.

Bypassing certificate errors

Certificate errors may occur with webview usage. The GDC allows you to bypass certificate errors temporarily or permanently.

When the application involves webview usage (such as with WebComponents, Single Sign-On (SSO), or the auto-logout prompt feature of the Genero Application Server), the Genero Desktop Client provides a mechanism for bypassing a certificate error. When a certificate error is encountered, a popup dialog displays the error and asks the user whether they want to bypass the error and load the page.
Figure 64: Certificate Error dialog

Table 54: Certificate error bypass options

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Bypass the specified certificate error from this host for the current GDC monitor session. This permission is stored in memory; restarting the GDC monitor removes this permission. After restarting the GDC monitor, any subsequent encounter of this certificate error from this host will cause the Certificate Error dialog to be displayed again.</td>
</tr>
<tr>
<td>Yes to All</td>
<td>Bypass the specified certificate error from this host, and create an entry in the $AppDataDir/hosts.xml file documenting the hostname, IP address, and certificate error. Adding the entry to the hosts.xml file makes it permanent. All future requests from this host with the same certificate error will also be bypassed and the request silently accepted. To retract this permission, edit the hosts.xml file and remove the entry. You can use the Manage known hosts dialog to remove the entry, or you can modify the hosts.xml file directly using a text editor.</td>
</tr>
<tr>
<td>No</td>
<td>The certificate error is not bypassed. The webview displays an error page.</td>
</tr>
</tbody>
</table>

Manage known hosts dialog

Access the Manage known hosts dialog from the Security tab, provided under the Options panel in the Genero Desktop Client. The GDC must be launched as administrator in order to access the Options panel.

The Manage known hosts dialog displays the contents of the $AppDataDir/hosts.xml file. From this interface, you can view known hosts and remove selected hosts.
Related concepts

Security configuration options on page 15
Access the Security tab to configure the security level, clear stored passwords, and view/clear known hosts.

GDC configuration file directories on page 18
The GDC configuration files are stored in two (default) directories: AppDataDir and AppCacheDir.

Microsoft® firewall configuration

By default, the Genero Desktop Client application is blocked from communicating through the Windows® firewall.

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, it can be changed in the Windows® Firewall settings.
Microsoft® User Account Control

Microsoft® User Account Control affects Genero Desktop Client.

The Microsoft® User Account Control 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.

Installation

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.

Runtime

Once GDC is installed, the Windows® Firewall will prompt the user to unblock the program, as described in Microsoft firewall configuration on page 138.

Although most of the features of Genero Desktop Client will work out of the box, some features will only work if you start GDC as administrator.

![Figure 66: Run as administrator](image)

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.
Front End Extensions

The Genero Desktop Client allows you to call external functions from your Genero program. You can create your own front-end extensions.

These functions are dynamically loaded by the front end when needed. To create your own extensions and use them from within your Genero program, or to learn more about the APIs provided for Windows® DDE support, Windows® COM support and the Windows® Mail extension, see the Front calls section of the Genero Business Development Language User Guide.

Legal notices

Genero Desktop Client 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).
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.

GDC configuration file directories

The GDC configuration files are stored in two (default) directories: AppDataDir and AppCacheDir.

Table 55: Configuration file directories on page 141 shows the locations of the default directories for the GDC configuration files.

Table 55: Configuration file directories

<table>
<thead>
<tr>
<th>Directory name</th>
<th>Directory location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppDataDir</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows®</td>
<td>C:\Users&lt;USERNAME&gt;\AppData\Roaming\Four Js\Genero Desktop Client&lt;VERSIONNUMBER&gt;\</td>
<td>Contains:</td>
</tr>
<tr>
<td></td>
<td>~.local/</td>
<td>• hosts.xml</td>
</tr>
<tr>
<td></td>
<td>share/Four Js/</td>
<td>• config.xml</td>
</tr>
<tr>
<td></td>
<td>Genero Desktop</td>
<td>• webcomponent default sub-directory</td>
</tr>
<tr>
<td></td>
<td>Client/&lt;VERSIONNUMBER&gt;/</td>
<td></td>
</tr>
<tr>
<td>Linux®</td>
<td>~.local/</td>
<td>• dictionaries sub-directory</td>
</tr>
<tr>
<td></td>
<td>share/Four Js/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Genero Desktop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Client/&lt;VERSIONNUMBER&gt;/</td>
<td></td>
</tr>
<tr>
<td>Mac®</td>
<td>~/Library/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Support/Four Js/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Genero Desktop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Client/&lt;VERSIONNUMBER&gt;/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or ~/Library/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Support/Four Js/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Genero Desktop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Client/&lt;VERSIONNUMBER&gt;/</td>
<td></td>
</tr>
<tr>
<td>Directory name</td>
<td>Directory location</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>AppCacheDir</td>
<td></td>
<td>Contains:</td>
</tr>
<tr>
<td>Windows®</td>
<td>C: \Users&lt;USERNAME&gt;\AppData\Local\Four Js \Genero Desktop Client\cache\</td>
<td>application cache</td>
</tr>
<tr>
<td></td>
<td>~/.cache/Four Js/</td>
<td>• images</td>
</tr>
<tr>
<td></td>
<td>Genero Desktop</td>
<td>• ftcache sub-directory</td>
</tr>
<tr>
<td></td>
<td>Client/</td>
<td></td>
</tr>
<tr>
<td>Linux®</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mac®</td>
<td>~/Library/Caches/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Four Js/Genero</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Desktop Client/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or /Library/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Caches/Four Js/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Genero Desktop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Client/</td>
<td></td>
</tr>
</tbody>
</table>

**Copyrights and Trademarks**

The Copyright and trademark page is shared amongst all Four Js documentation. Appearance in this page does not guarantee a mention within the current document.

**Copyright notice**

© Copyright Four Js Development Tools, Inc. Use, duplication or disclosure restricted.

**Trademarks**

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated.

IBM and WebSphere are trademarks or registered trademarks of International Business Machines Corp., 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.

Linux is a registered trademark of Linus Torvalds.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Other company, product, or service names may be trademarks or service marks of others.
Index

A
Abstract User Interface (AUI) tree
  inspecting the AUI tree 42
accessibility
  limitations 45
applications
  connections panel 31
  debug panel and logging system
    debug system 36
    logging system 35
  login box 29
  shortcuts
    environment variables 29
    shortcut management 28
    shortcut wizard
      direct connection 21
      HTTP connection 25
      local execution 27
  terminals panel 33
AUI tree
  inspecting the AUI tree 42
auto-update 45

C
compatibility
  FGL 6
  GAS 6
configuration
  additional configuration file 18
  advanced options 9
  configuration file directories 18, 141
  connections options 13
  desktop shortcut 6
  macOS 6
  Preferences options 7
  report options 17
  security options 15
  starting the GDC 6

D
debug
  WebComponents 48
dictionaries 49

E
Extensions
  Front end extensions 140

F
features
  command line
    examples 41

G
general information
  What is GDC 5
Glossary
  Security terms 100

L
legal notices 140

S
security
  firewall configuration 138
  security levels 98, 136
security - GDC and SSH
  overview 102
  prerequisites 102
  simple setup 104
security - Microsoft
  user account control 139
security - port forwarding and firewalls
  client-side firewall 111
  port forwarding 105, 111, 114
  server-side firewall 114
security - Secure Server implementation
  changing passwords 129
  expired passwords 128
  overview 121
  password management 128
  prerequisites 121
  setup SSH login 122
  setup telnet 126
  shell script 122
security - SSH configuration troubleshooting
  changing the GDC port 136
  session expiring 136
  wireless systems 136
spellchecking 49

T
TextEdit
  spell checker 83

U
upgrade guide
  2.2x 93
  2.3x 91
  2.4x 90
2.5x 90
3.0x 89
3.10 88

W

what's new
  Version 2.20 79
  Version 2.21 73
  Version 2.22 70
  Version 2.30 59
  Version 2.32 58
  Version 2.40 52
  Version 2.50 51
  Version 3.00 50
  Version 3.10 5, 50

Windows
  Security Blocking 19