Genero Java Client

User Guide
Version 2.00

Note: This documentation is for Genero 2.00. See the corresponding on-line documentation at the Web site **http://www.4js.com/online_documentation** for the latest updates. Please contact your nearest support center if you encounter problems or errors in the on-line documentation.

Genero Java Client

Table Of Contents

General

Applications

Features

Applet/Java Web Start

Security

General Overview

Topics:

- Presentation
- Requirements

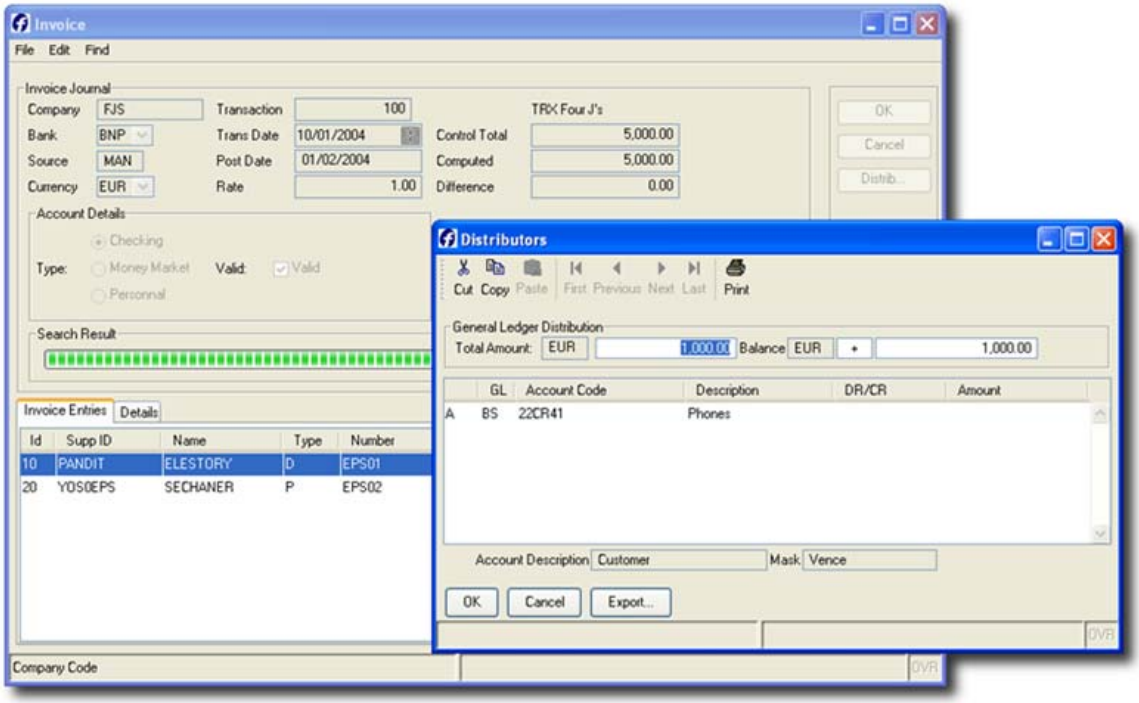
Presentation

Genero Java Client (GJC) is a graphical Front End for Genero Runtime System. Genero Java Client is written in Java and can be run under any operating system which supports J2SE Java Runtime Environment (JRE) version 1.4.2 or higher. Such systems include:

- Microsoft Windows systems
- Apple Mac OS X (10.2 or higher)
- Linux

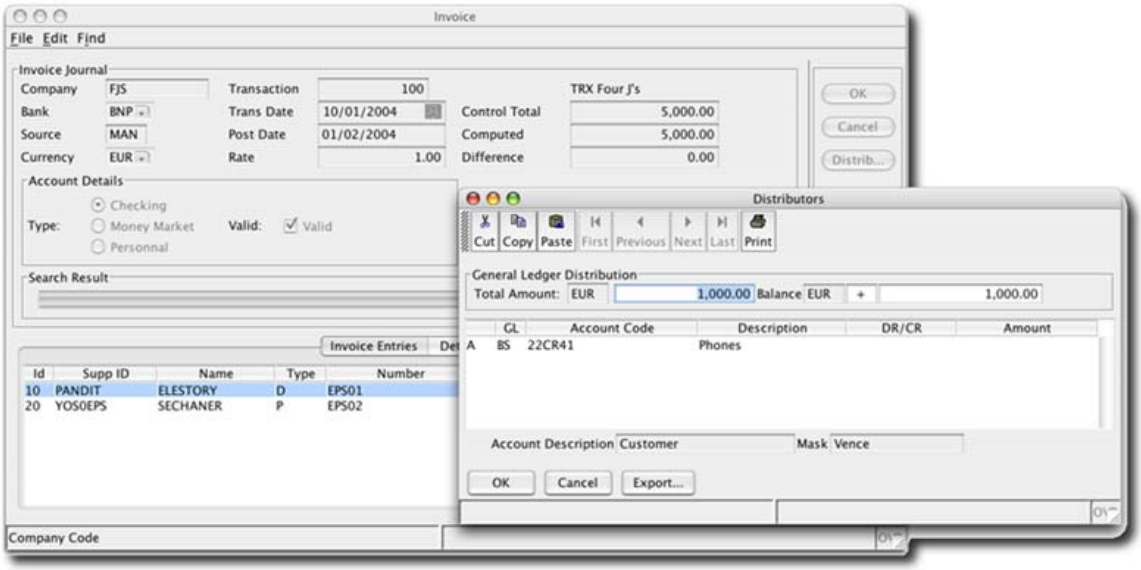
GJC can also be embedded into an HTML page thanks to an Applet, which will work on any browser that supports Java Plug-in technology with a 1.4.2 or higher J2SE Java Runtime Environment. The following screen shots illustrate the same application running on different platforms:

Microsoft Windows XP version:

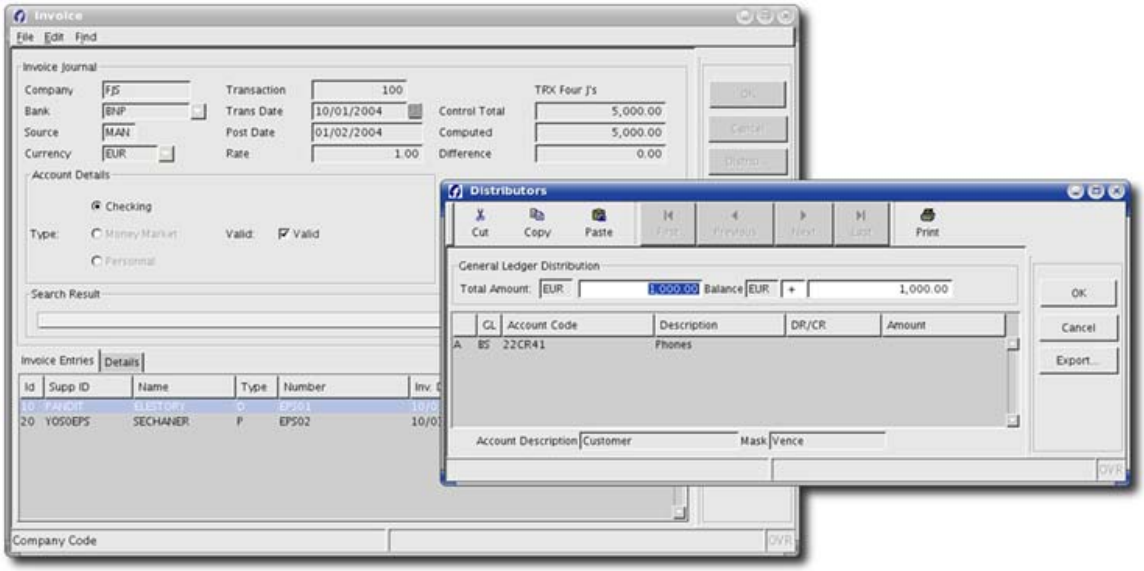


Genero Java Client

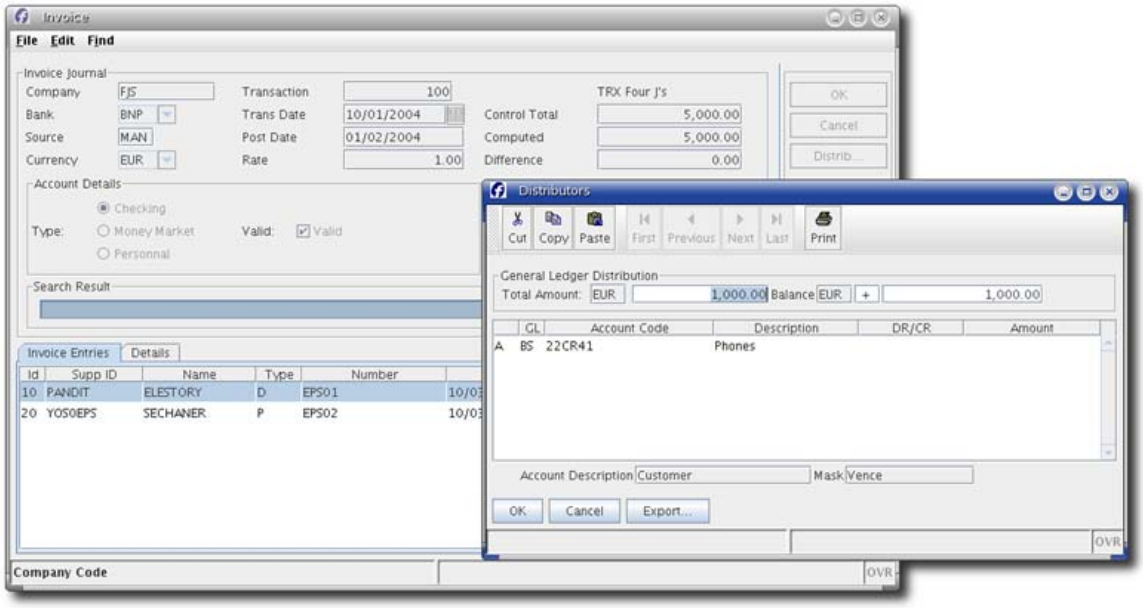
Apple Mac OS X 10.3 version:



Linux Version (using GTK look-and-feel):



Linux Version (using Metal look-and-feel):



Requirements

All systems that support J2SE Java Runtime Environment version 1.4.2 or higher can run GJC. However, as the Java environment may require a certain amount of CPU power and memory, the minimum hardware should be:

- 600 Mhz processor
- 512 MB of RAM
- 5 MB of available disk space for GJC and 72 MB for JRE

Installation

The installation process installs the Front End on the Workstation.

Summary:

- Prerequisites
- Windows Systems
- Mac OS X and X11 Systems

Prerequisites

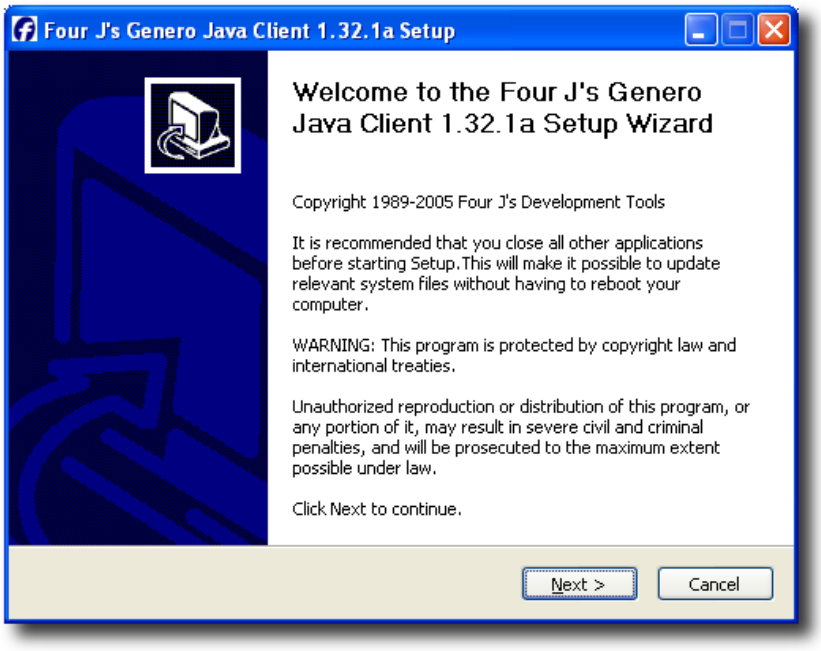
Genero Java Client (GJC) works only with a Genero Runtime System. It is not compatible with Four J's BDL Runtime System.The GJC installer, as with any Java-enabled software, requires that J2SE Java Runtime Environment (JRE) version 1.4.2 or higher be installed.

Windows Systems

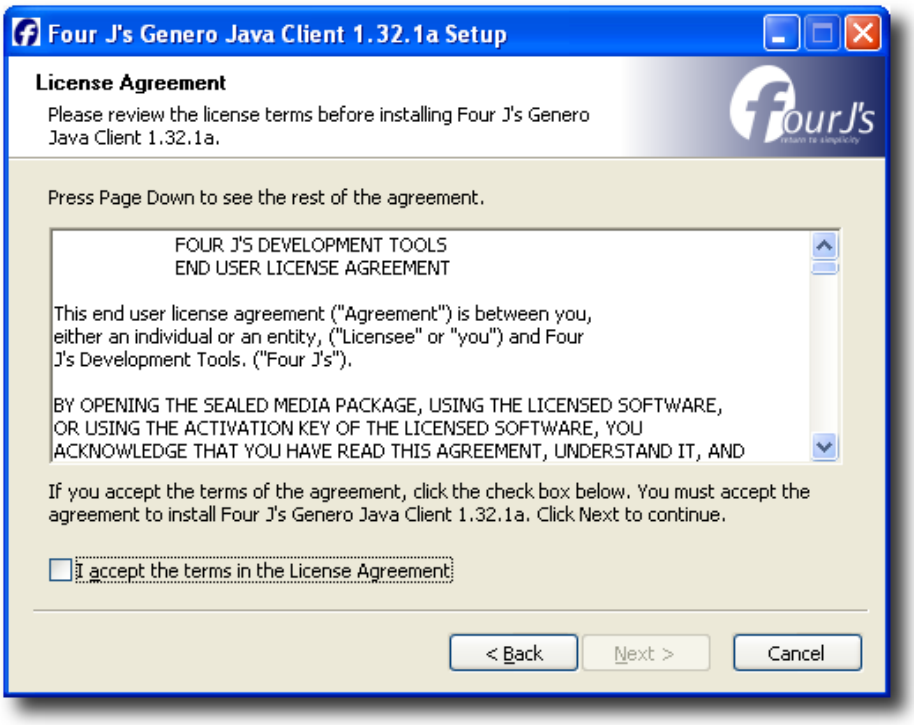
To start the installation:

1. Close all applications.
2. Execute the installation program.

Welcome screen- Press "Next" to start the installation.

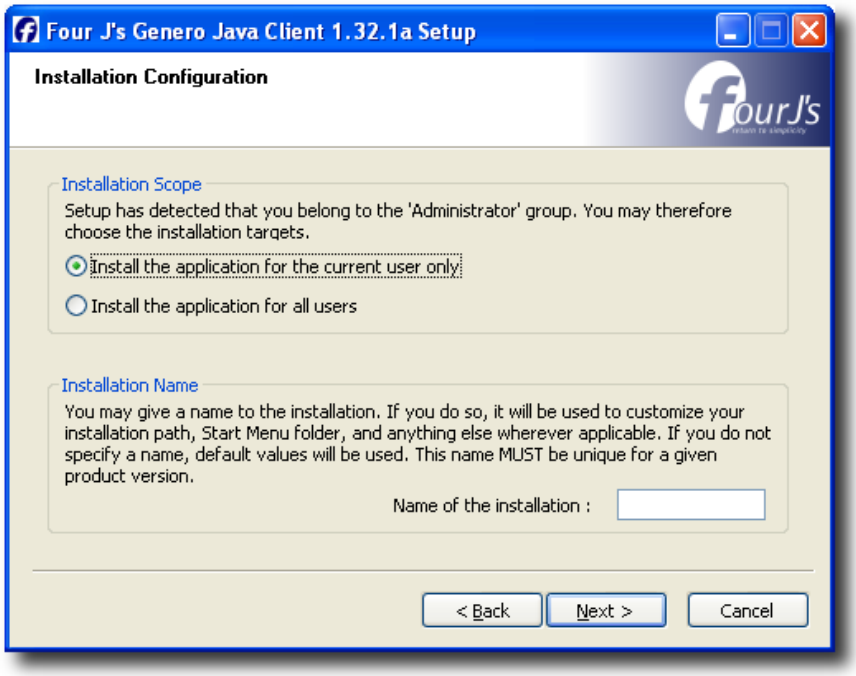


Licenses agreement:



- Select "I accept the term of this license agreement".
- Click "Next".

Installation configuration:

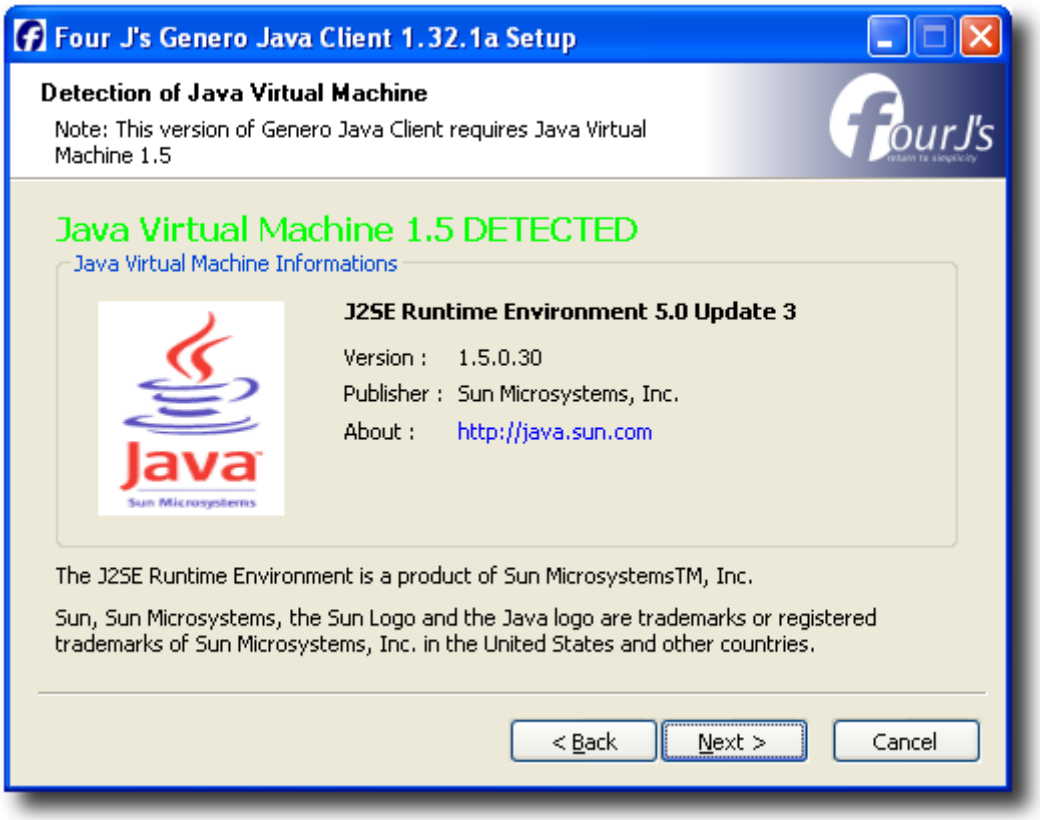


- Depending on your system, the installation system will ask you if you want to install GJC for the current user only or for all users. Select the appropriate choice.
- If desired, provide an installation name. You can leave this field blank, a default installation name will be used.
- Click "Next"

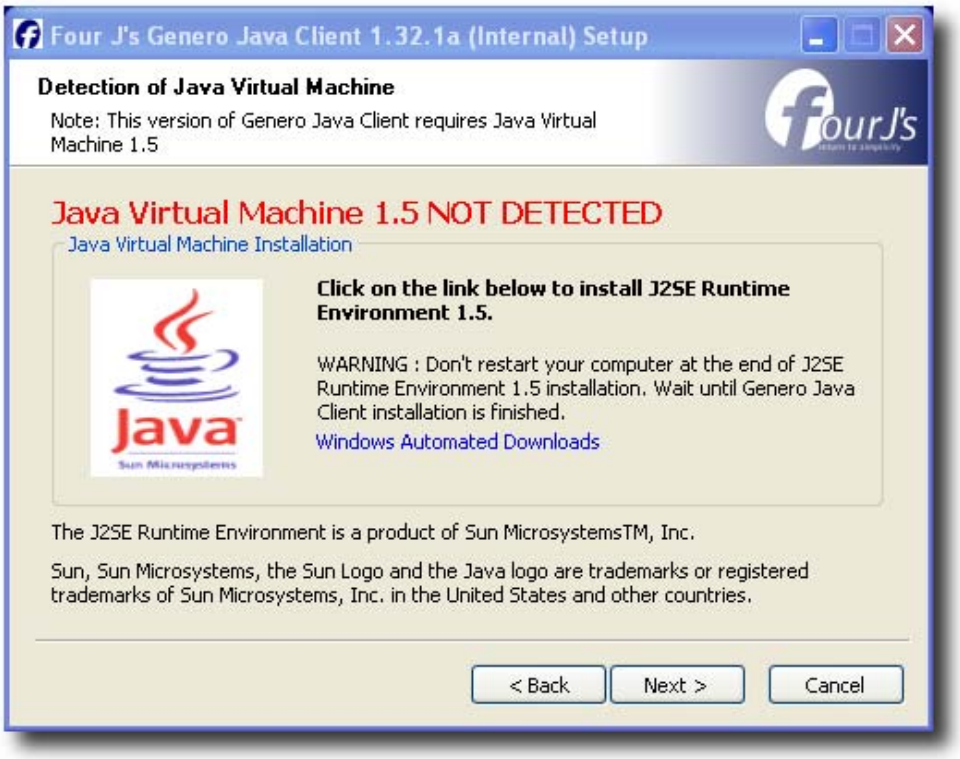
Detection of Java Virtual Machine

- check if JVM is installed.

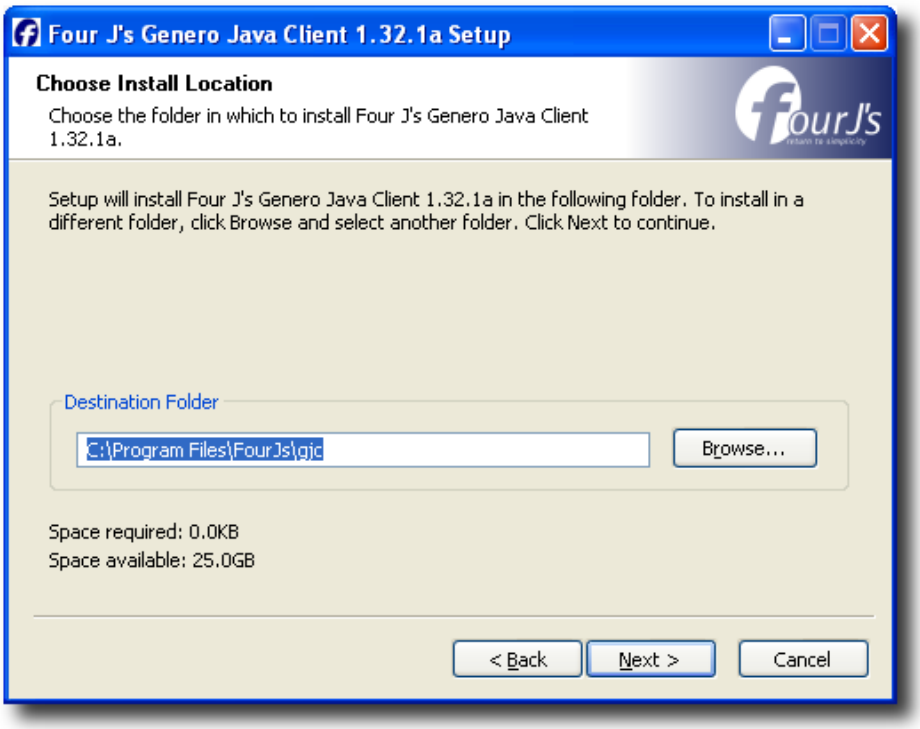
If the Java Virtual Machine is already installed, only press "Next" to continue the installation:



If the Java Virtual Machine is not installed, click on "Windows Automated Downloads" link and follow instructions to install the lasted Java Virtual Machine:

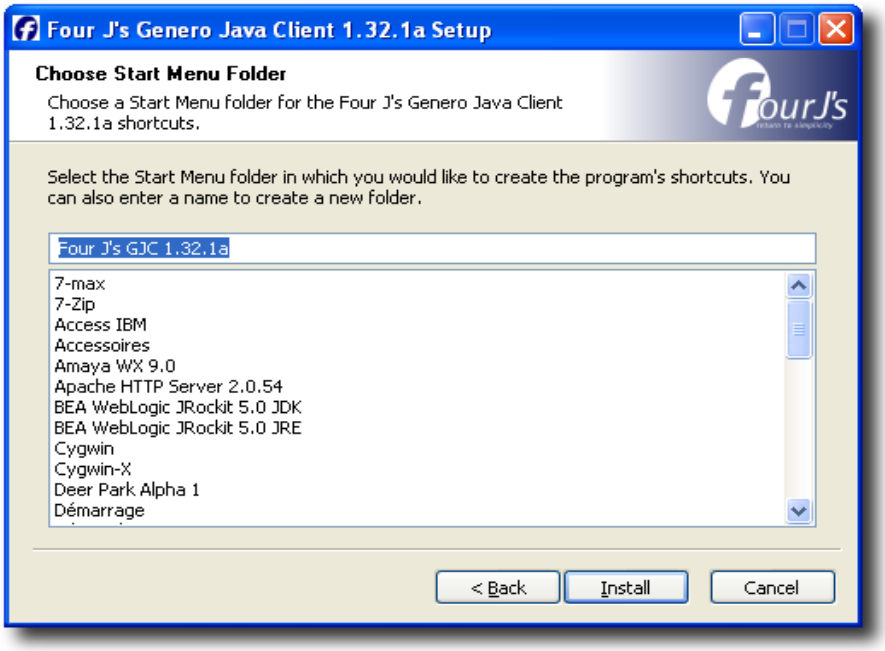


Choose Install Location:



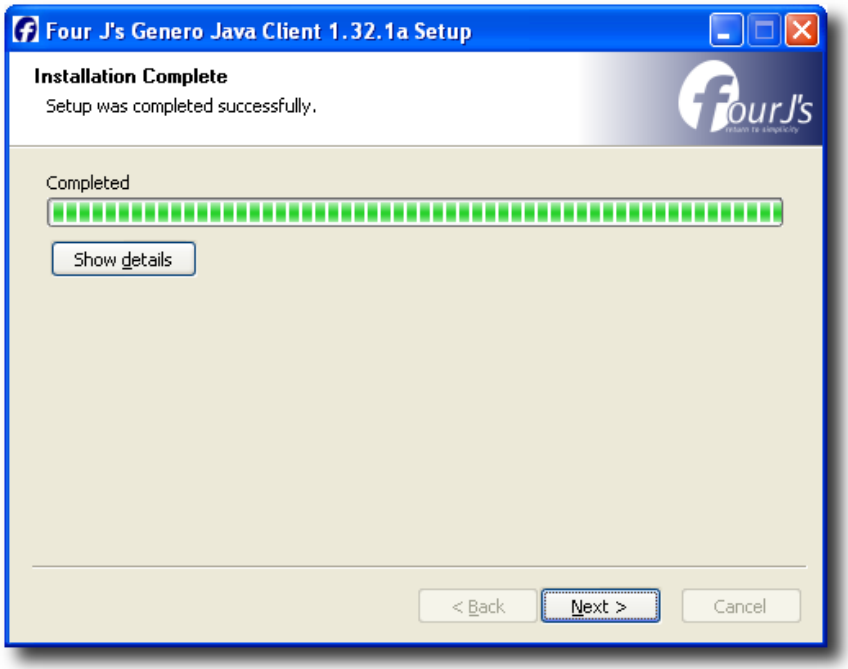
- Select the folder in which you want to install the GJC.
- Click "Next"

Choose Start Menu Folder:

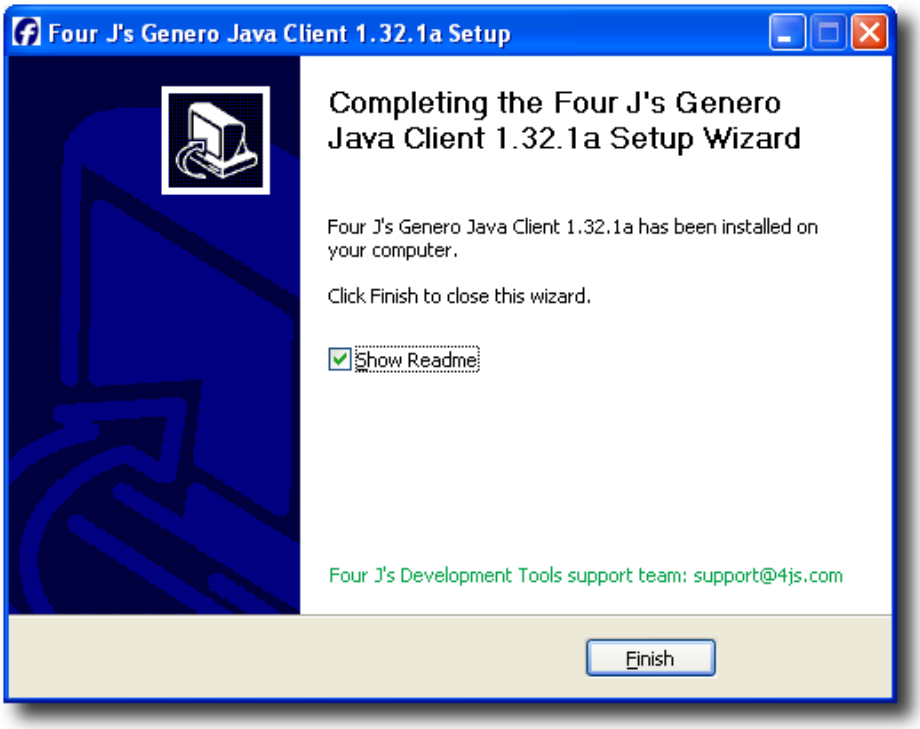


- Select the StartMenu folder in which GJC shortcuts will be set.
- Click "Install"

Finishing the installation:



- When all the required files are copied, click "Next".



- You can now close the installer by pressing "Finish". You should now be able to launch Genero Java Client

---

Mac OS X and X11 Systems

Launching the installer

The installation process is provided by a executable Java Archive (JAR) file. If JAR files are correctly associated with the JRE, a simple double-click on the installer jar should launch the installation of Genero Java Client. Otherwise, the installer can be run by the following command:

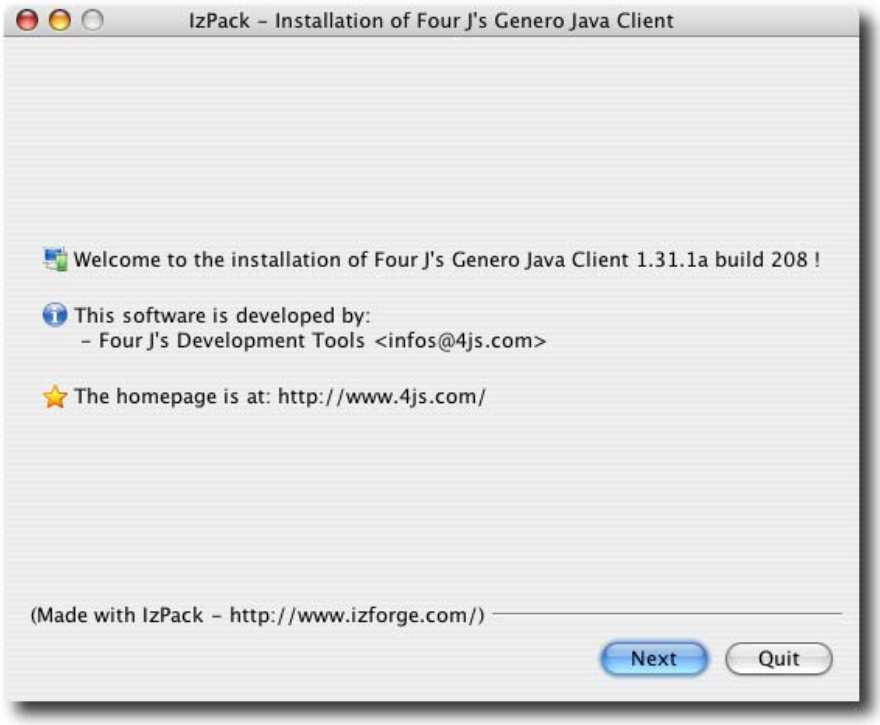
```
java -jar gjc-installer.jar
```

Installer step

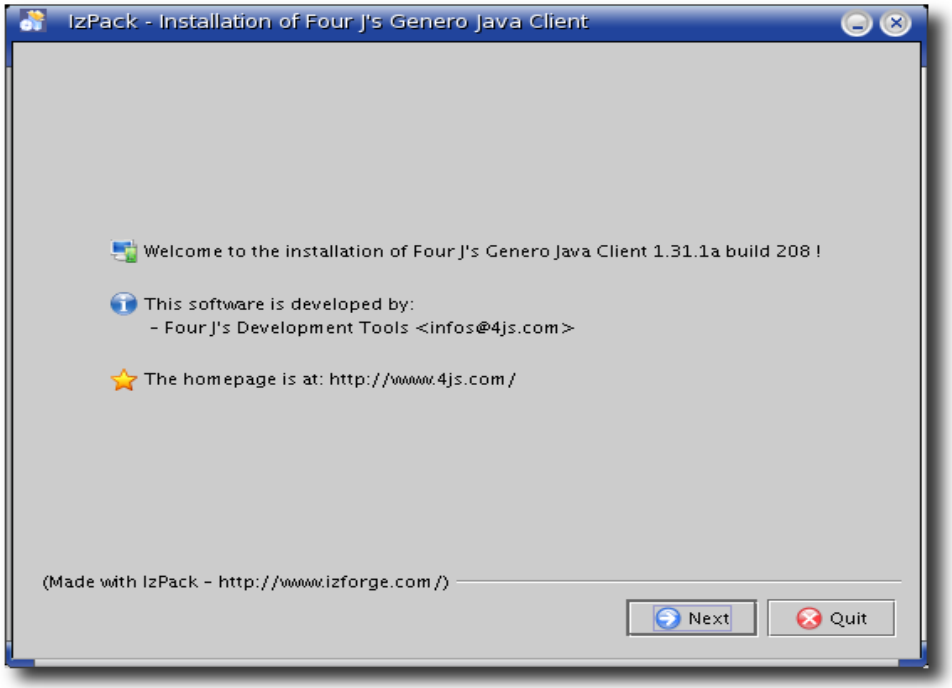
First of all, because the installer is a java installer, the different installation steps will be the same on any supported platforms. However, be aware that the dialog look-and-feel can change slightly, depending of the underlying operating system. The following screenshots come from different operating systems (Apple Mac OS X 10.3 and Linux) and do not represent all possible displays.

Welcome screen: Press "Next" to start the installation.

- Apple Mac OS X 10.3

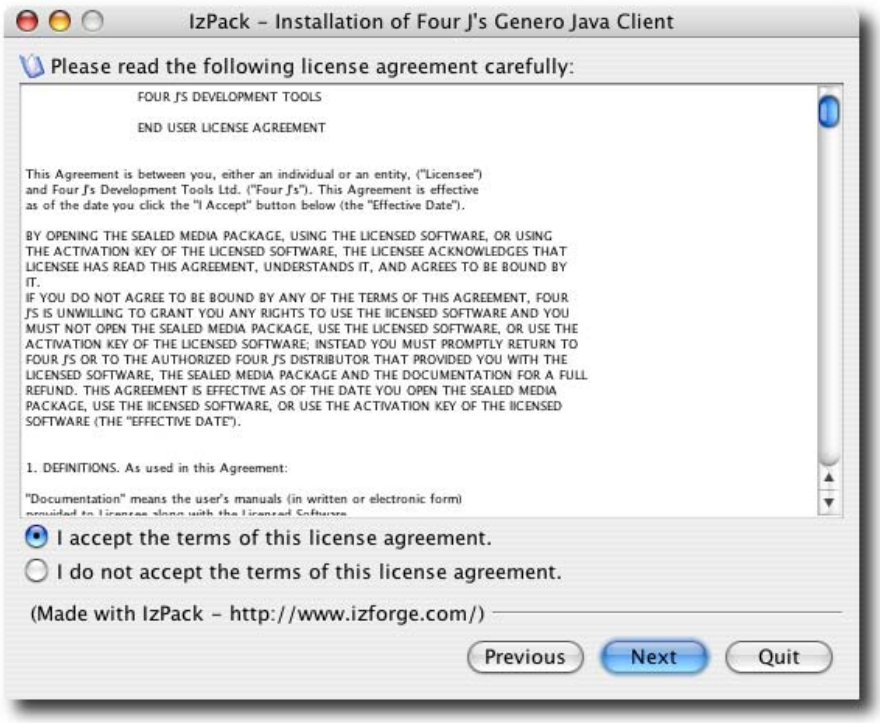


- Linux with KDE desktop

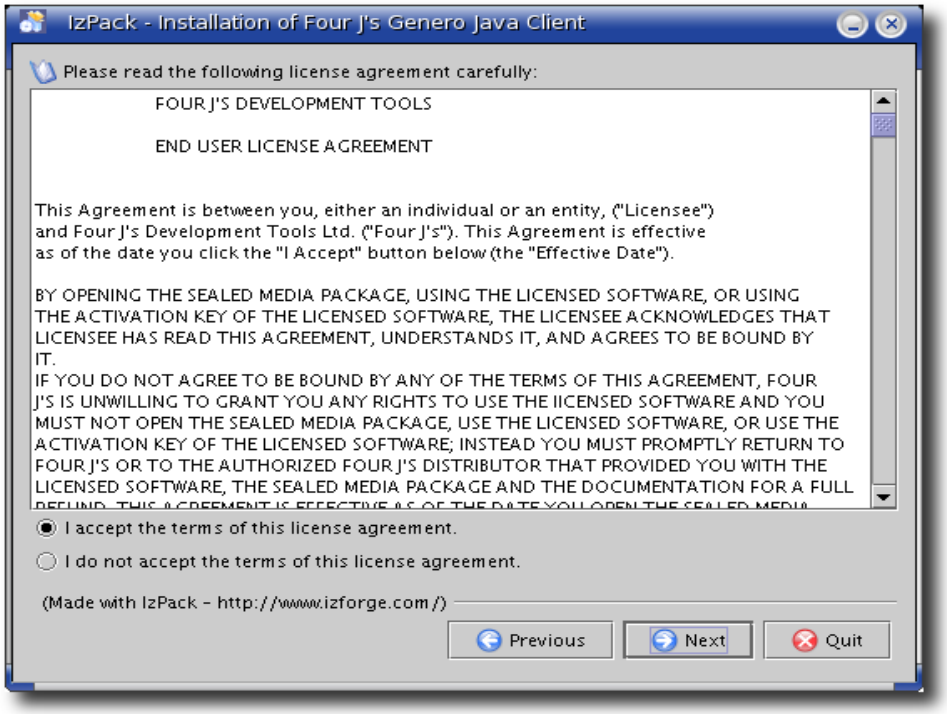


Licenses agreement: Select "I accept the term of this license agreement" if you accept it to continue the installation.

- Apple Mac OS X 10.3

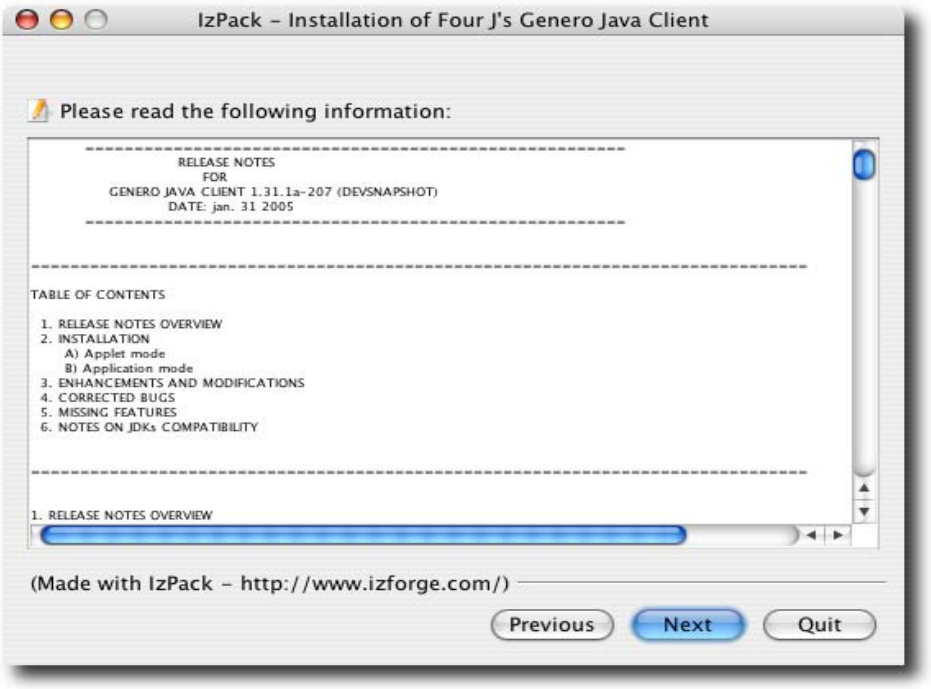


- Linux with KDE desktop

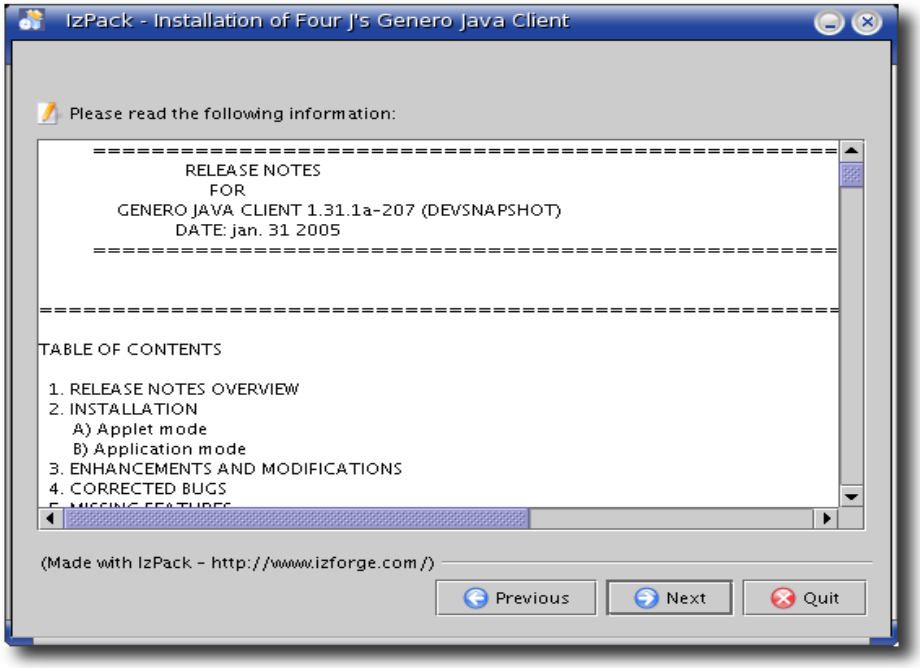


Release and installation notes: Read the installation and release notes.

- Apple Mac OS X 10.3

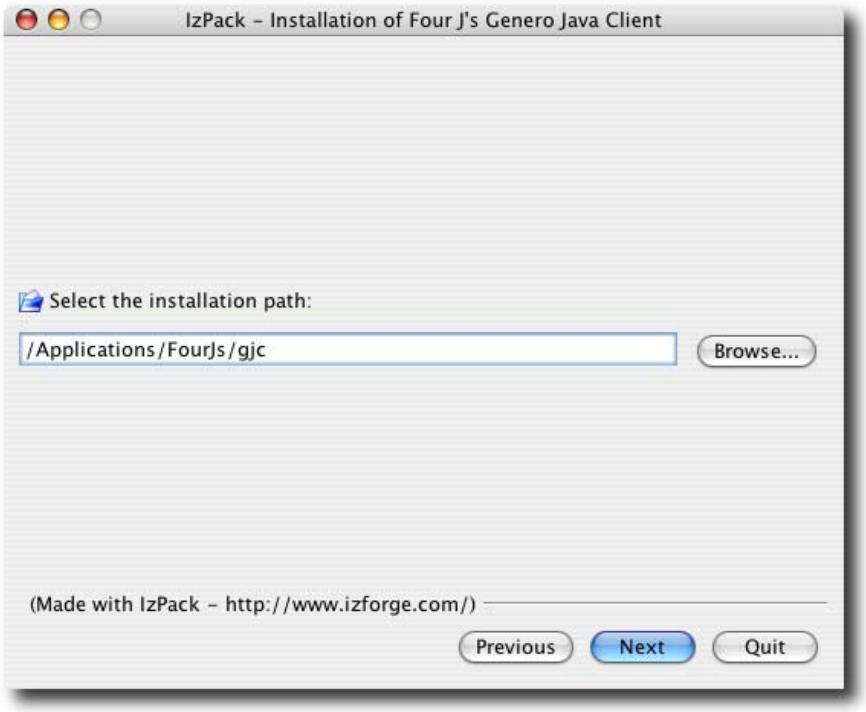


- Linux with KDE desktop

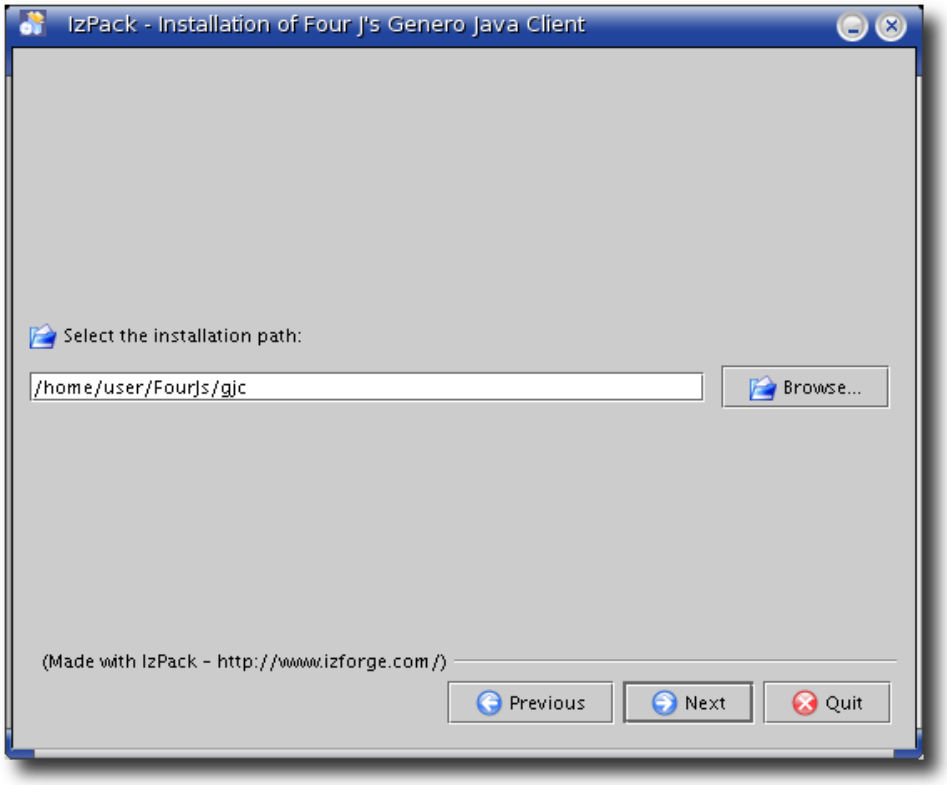


Location selection: Choose the location where do you want to install the GJC.

- Apple Mac OS X 10.3

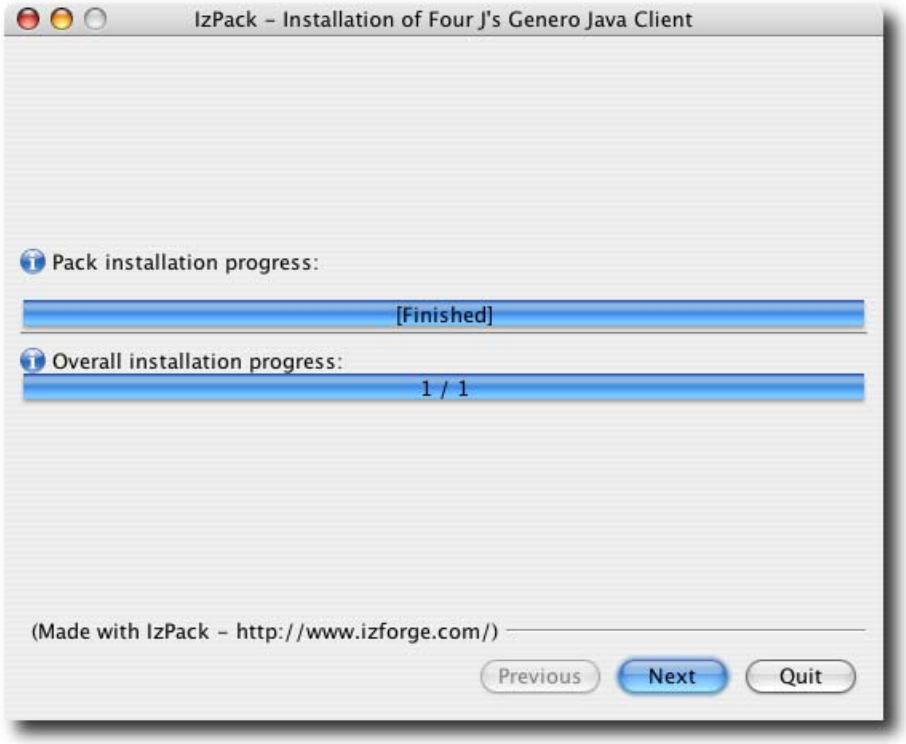


- Linux with KDE desktop

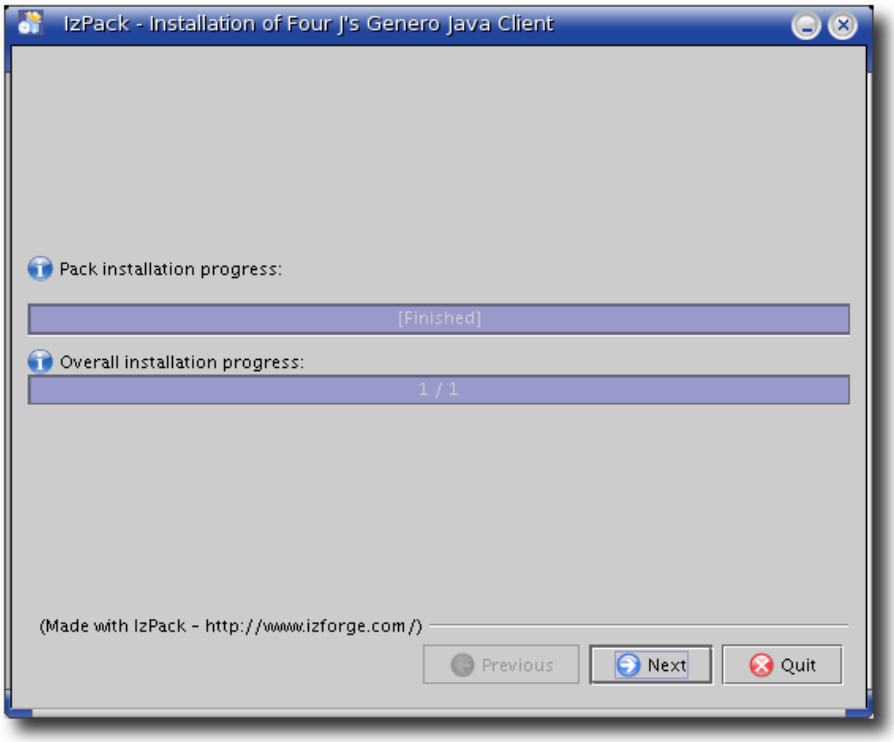


File copying: When all the required files are copied, just click "Next".

- Apple Mac OS X 10.3

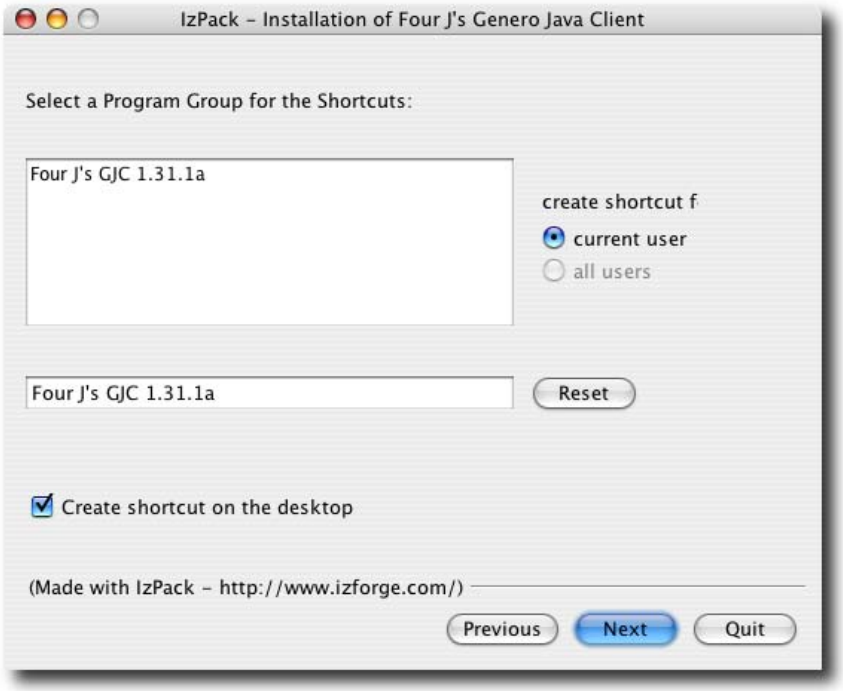


- Linux with KDE desktop

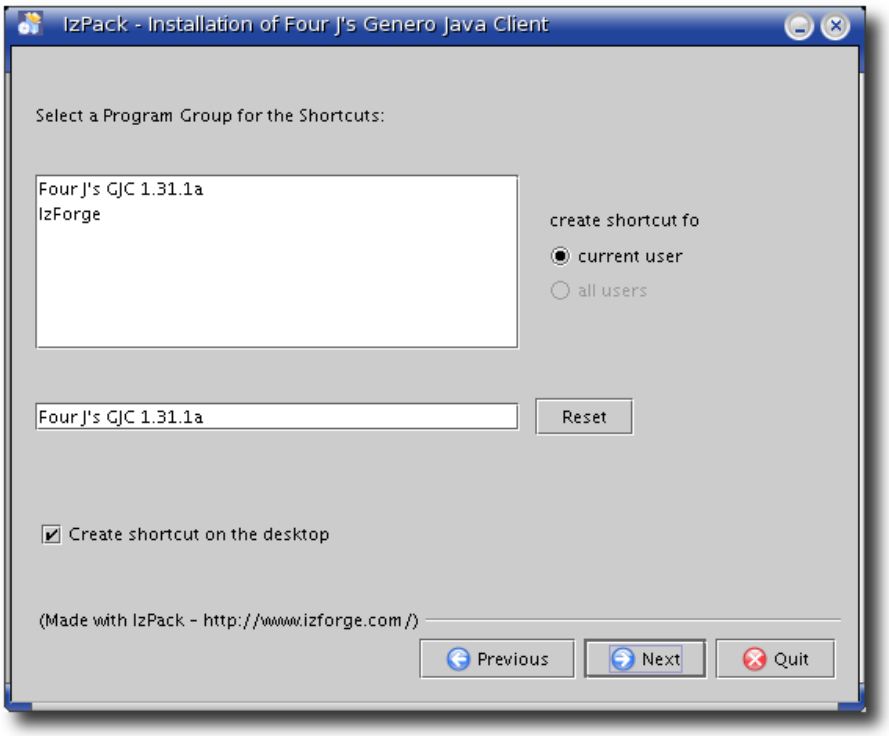


Shortcut creation: This step allow you to create a shortcut in a program group and/or on the desktop. Your user must have administrative rights to create and modify system files; otherwise the shortcut will not be created even if the installation process succeeds.

- Apple Mac OS X 10.3

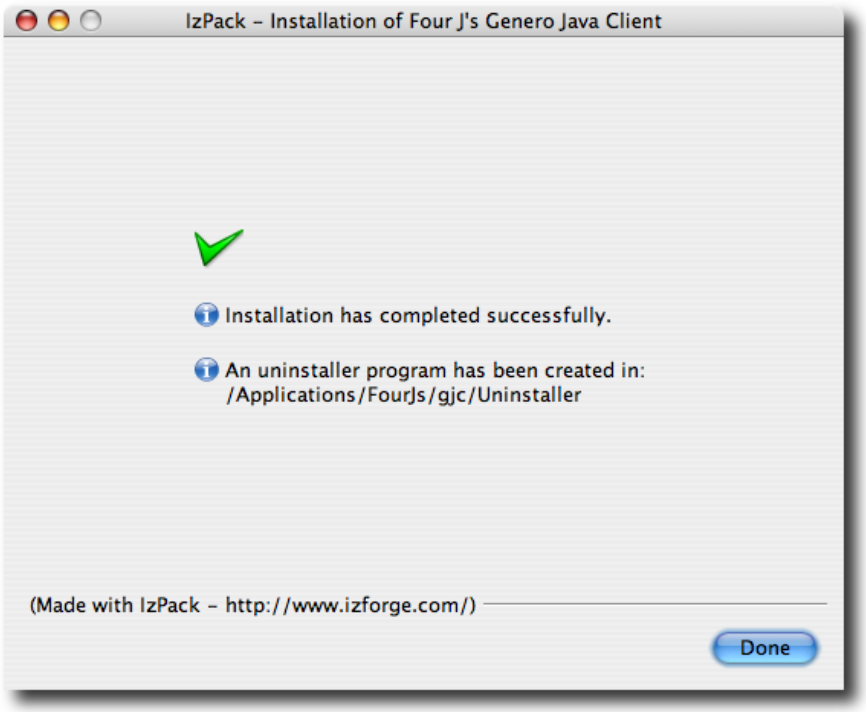


- Linux with KDE desktop

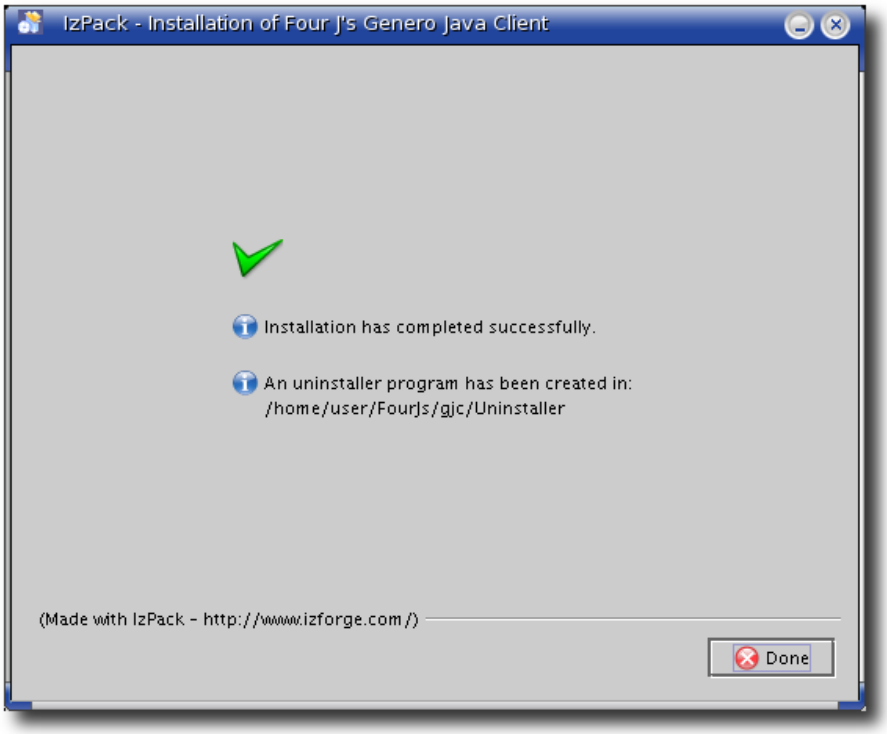


Finishing the installation: You can now close the installer by pressing "Done". You should now be able to launch Genero Java Client.

- Apple Mac OS X 10.3



- Linux with KDE desktop



Starting and Configuring Genero Java Client

Summary:

- Starting GJC
- Configuring GJC

Starting GJC

If a shortcut was successfully created by the installer, a double-click on the shortcut launches the GJC. You can also start GJC with the following command:

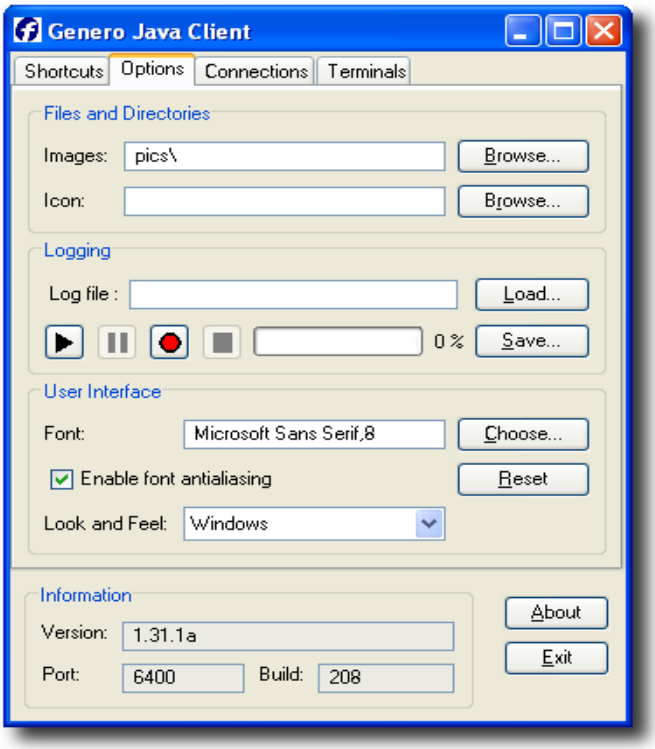
```
java -jar gjc.jar
```

Your working directory must contain the **gjc.jar** file, or you must change the command to include the location of the file. Usually, the **gjc.jar** file is located in the **bin** directory where GJC was installed.

By default, GJC listens for Runtime System connections on port 6400. You can specify an alternate port by starting GJC with the "`-p`" option. If the port is not available, GJC tries the next port, stopping when it finds the first available port.

For a list of all command line options, refer to the Command Line section.

Configuring GJC



The following options may be configured:

- Local image path
- Default icon
- Default font
- Font antialiasing
- Look and feel

Path to local images:

> Specify the directory path where GJC should look when an image is needed. When an image is required, GJC will first check if the given name corresponds to an absolute file name, then it will look into the path you have specified here, then in /pics directory. If it has still not found the image, it will draw a "..." picture.

Default Icon:

> Specify the default icon to be used for GJC. This icon is used for the taskbar, shortcuts, the terminal, and applications.

Default Font:

> Specify the default font for GJC. This font is used everywhere in your applications.

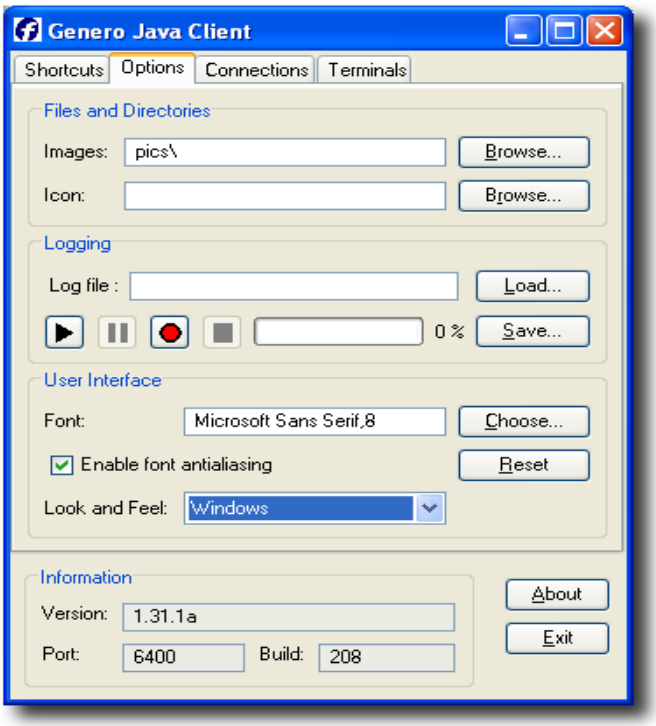
Font Antialiasing:

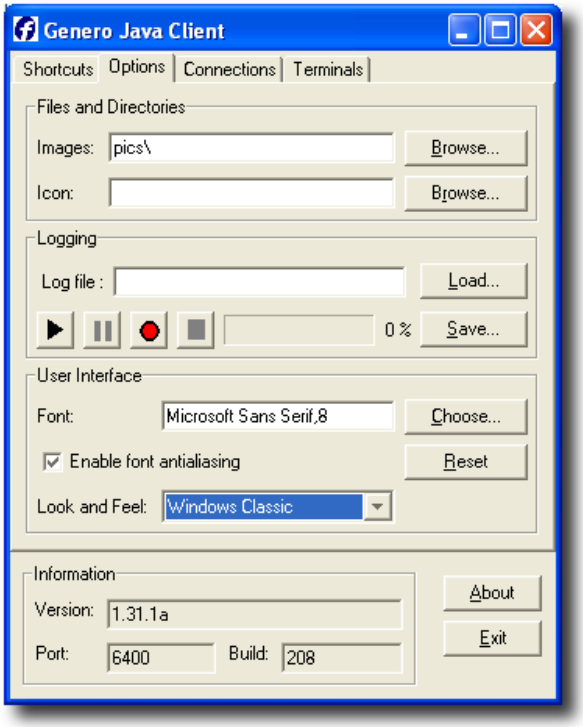
If checked, font antialiasing or font "smoothing" is enabled. When enabled, it provides a better look but worst graphical performances. The performances loss are correlated with your hardware configuration (CPU, graphic card, and so on), so it is difficult to estimate the extent of the performance degredation. Four J's recommends you enable this option and test the performance against a large table. If you do not see any major flickering, you can leave this option enabled.

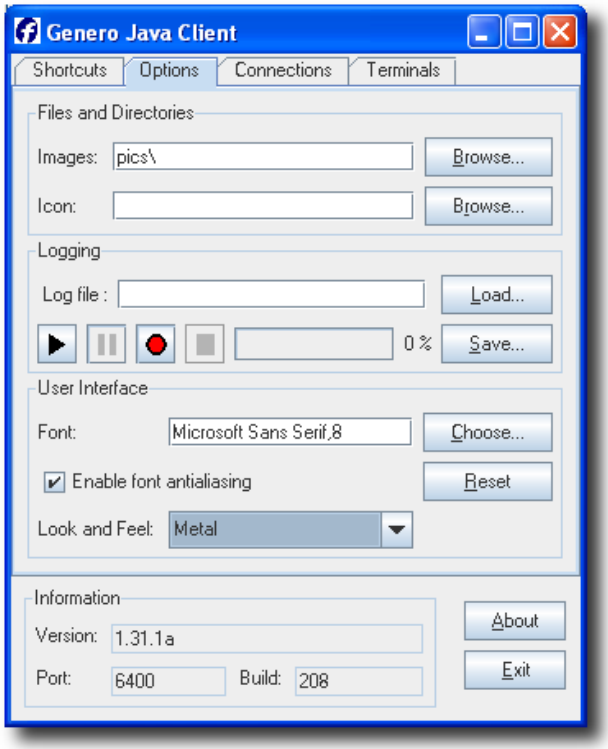
Look and Feel:

Specify the look-and-feel of the application. A look-and-feel is a set of graphical properties which define the graphical design of the form, the widget, the label, and so on. This setting can only be changed when no Genero applications are running. Because some of the available look-and-feel are system-dependent, the list of available look-and-feel settings may differ from one operating system to another. For example, on Microsoft Windows XP you can select "Windows Classic" (a.k.a. Windows 2000 theme), "Windows" (a.k.a. Windows XP theme), and "Metal" look-and-feel; while on Linux you can select either the "Metal" or "GTK" look-and-feel.
Screen shots of the GJC monitor using different look-and-feel settings on different platforms are illustrated below.

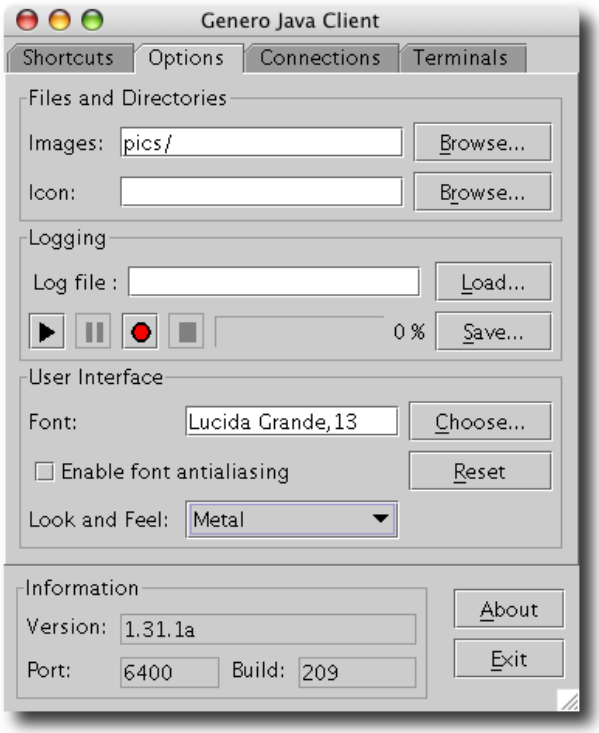
Microsoft Windows available look-and-feel:



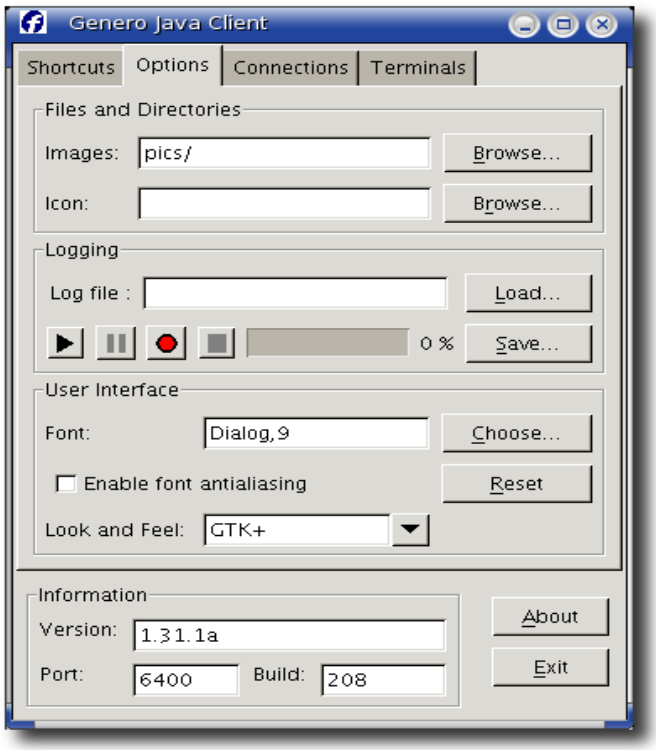

Genero Java Client
Shortcuts
Options
Connections
Terminals
Files and Directories
Images:
pics\
Browse...
Icon:
Browse...
Logging
Log file :
Load...
0 %
Save...
User Interface
Font:
Microsoft Sans Serif,8
Choose...
Enable font antialiasing
Reset
Look and Feel:
Windows Classic
Information
Version:
1.31.1a
Port:
6400
Build:
208
About
Exit

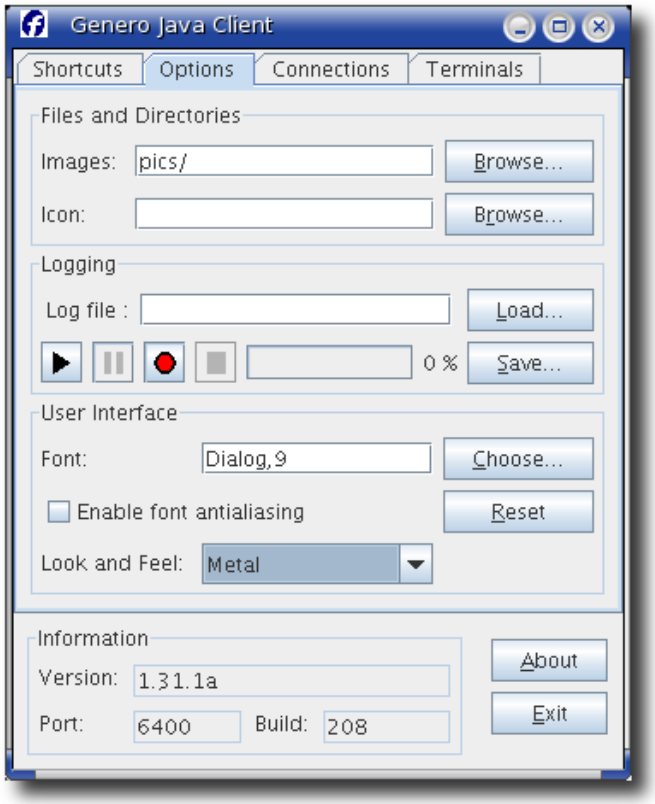

Genero Java Client
Shortcuts
Options
Connections
Terminals
Files and Directories
Images:
pics\
Browse...
Icon:
Browse...
Logging
Log file :
Load...
0 %
Save...
User Interface
Font:
Microsoft Sans Serif,8
Choose...
Enable font antialiasing
Reset
Look and Feel:
Metal
Information
Version:
1.31.1a
Port:
6400
Build:
208
About
Exit

Apple Mac OS X 10.3 available look-and-feel



Linux with KDE Desktop available look-and-feel





Logging System

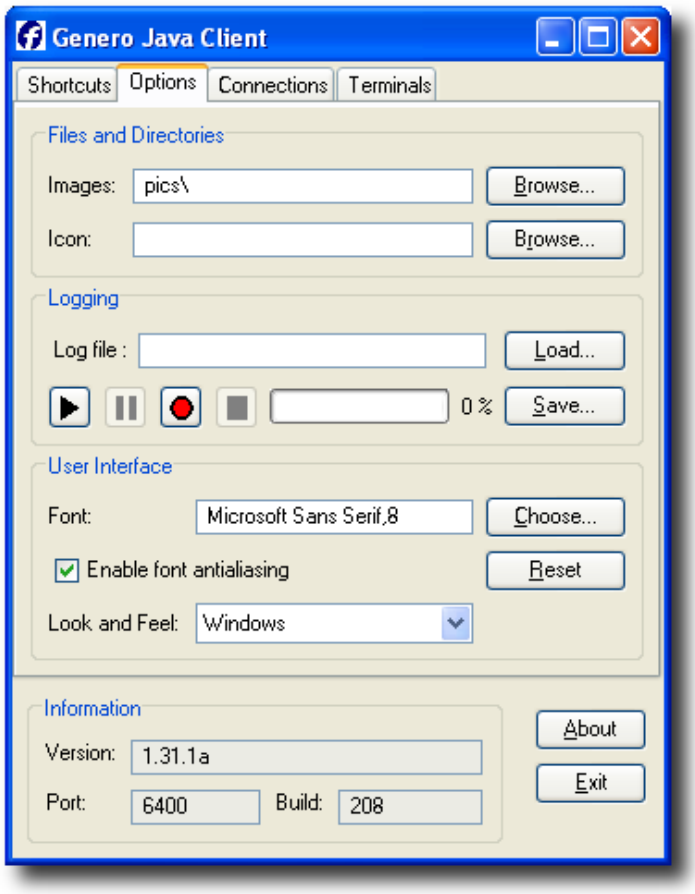
Summary:

- Overview
- Record a Scenario
- Replay a Scenario

Overview

GJC provides a logging system. This system will help you to:

- Debug your applications
- Create a demo



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

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

Record a Scenario

To record a demo:

1. Select a log file to store the scenario.
2. Click on the Record button to start recording.

WARNING! Only those applications launched AFTER recording starts are stored.

Replay a Scenario

To replay a demo:

1. Select the log file where the scenario is stored.
2. Click on the Play button.

The scenario starts. You can pause the replay by clicking on the Pause button. The progress bar indicates the progress of the demo.

WARNING! User interaction is not possible when replaying a scenario. You may have stop recording the demo before the end of the application. In this situation, to kill this application, you must use the Connections panel.

GJC F.A.Q.

Contents:

---

1. General Questions

1.1 What does GJC mean ?

GJC stands for Genero Java Client.

.

1.2 What is the Debug Console ?

The Debug Console shows you the communication between the Front End and the Runtime System. See Debug Console doc. for more details. You can open this window by clicking on "Console" button on the Connections Panel.

.

1.3 I did control-right-click on a form, and a strange window appeared ! What's this ?

You have opened the 'Debug Tree'. This window shows all the Abstract User Interface Tree and the nodes attributes. This window can only appear when the GJC is started in debug mode.

If you are an end-user, you do not care about this window. If you are a programmer, please refer to Genero's documentation for more information about the AUI Tree.

.

1.4 How can I change the default icon ?

GJC allows you to change the default icon used in:

- the task bar
- the shortcuts system (default icon)
- the terminal system (default icon, Windows systems only)
- each application (default icon)

You can refer to the Configuring GJC section to see how to specify this icon.

.

1.5 What does GJC do when it has to display an image ?

GJC uses the following algorithm to search for an image:

Step	Test done	Action	Example
1.	filename starts with "http://"	uses the HTTP protocol to get the image from a web server	http://www.4js.com/fourjs/site/img/template/logo.jpg
2.a.	filename corresponds to a file		"c:/mypictures/logo.jpg" "../../app/pics/img2.bmp"
2.b	filename + extension correspond to a file		"c:/mypictures/logo.jpg" --> c:/mypictures/logo.jpg "../../app/pics/img2" --> ../../app/pics/img2.bmp
3.a.	look into <userdir> if filename exists		userdir = "c:\mypictures" "logo.jpg" -->>c:\mypictures\logo.jpg
3.b	look into <userdir> if filename + extension exists	uses the given file	userdir = "c:\mypictures" "logo" --> c:\mypictures\logo.jpg
4.a.	look into <gjc.jar> if filename exists		"smiley.jpg" --> <gjc.jar>/pics/smiley.jpg
4.b	look into <gjc.jar> if filename + extension exists		"smiley" --> <gjc.jar>/pics/smiley.jpg

GJC supports the following extensions : bmp, gif, png, ico, jpg.

1.6 What are these files in the etc directory ?

The *<GJCDIR>/etc/* contains some files that are created when needed.

FileName	Description
`config.xml`	The main configuration files ; contains GJC parameters, and non "local" shortcuts.
`hosts.xml`	Contains the list of authorized hosts

1.7 I plan to install Windows XP Service Pack 2. Is there any known problem with GJC ?

Changes related to Windows XP SP2 are described in this section.

1.8 How do I install java plugin on Mozilla/Firefox browsers with Linux ?

At present, there is no automatic installation for java plugin on Mozilla and Mozilla Firefox.

On Linux, Mozilla **requires** JRE 1.4.2 or later.
It is recommended that you use Java Runtime Environment 5.0 or later if possible.

Mozilla 1.4 and later, and Mozilla Firefox, are compiled with gcc 3.x. A gcc 3.x compatible version of the Java plugin must be used. JRE 1.4.2 and later contain a compatible plugin.

If you installed the Java Runtime Environment 5.0 Update 3, this plugin is /usr/java/jre1.5.0_03/plugin/i386/ns7/libjavaplugin_oji.so - and to install it for Mozilla (including Mozilla Firefox), do the following:

- Open a terminal
- Change to your Mozilla (or Mozilla Firefox) plugins directory
- Issue the following command: **ln -s /usr/java/jre1.5.0_03/plugin/i386/ns7/libjavaplugin_oji.so ./libjavaplugin_oji.so**

If you install your JRE in a different way (self extracting package, debian package, etc), libjavaplugin_oji.so will quite likely be in a different location. Do not just blindly use the command listed above!

In JRE 1.4.2, this file was in plugin/i386/ns610-gcc32

Always make a **symbolic link**, as shown above, instead of copying the plugin. If you copy the plugin, your browser will crash every time you open a page containing a Java applet.

If you are using an older Linux distribution, you may need to install the gcc3 support libraries, as the gcc 3.2 version of the Java plugin requires libgcc_s.so.1 to operate.

If you are using an old or unofficial build of Mozilla (1.4a or later) or Mozilla Firefox, you can check which compiler was used by entering about:buildconfig in the location bar and pressing enter. You will see a line such as "gcc version 3.3.2", which will show the compiler that was used. If gcc2.9x was used, you need to use the ns7-gcc29 or ns610 plugin, not the ns7 or ns610-gcc32 plugin.

1.9 Why doesn't Java Web Start work on Mozilla/Firefox browsers with Linux ?

This is because it is not possible to detect MIME type **application/x-java-jnlp-file** on Gecko-based browsers.
To allow your browser to be able to recognize the **.jnlp** extension, you need to do it manually.

In your Browser,
go to Edit->Preferences->Navigator->Helper-Applications->New Type
Fill fields like follow:
MIME type : application/x-java-jnlp-file
Description : Java Web Start
Extension : jnlp

Open it with : $JAVA_HOME/bin/javaws

---

2. Shortcut Problems

2.1 I can't see any shortcut even if I create new shortcuts; is this normal ?

No, of course not. It should not happen, but it may be that the file where the shortcuts are saved is corrupted. This file can be found in *<GJCDIR>/etc/config.xml*. The best solution is to remove it and re-create your shortcuts. You can also edit it, but this operation is reserved for experienced users.

WARNING! This file is automatically created by GJC and should not be changed directly. Changes may introduce uncontrollable problems.

.

2.2 When starting a direct shortcut, I got "Error #1, Host not found". What does it mean ?

It means that the host you have specified when creating the shortcut with the Shortcut Wizard can't be reached from your computer. Try to ping the host using the *ping* command. If this command fails, check your network configuration or contact your system administrator. Also check the computer host in your shortcut configuration.

.

2.3 I'm under Windows, I've started a direct Shortcut, and nothing happens. What's wrong ?

First, wait a few seconds. The Naming Resolution System can take a couple of seconds and throw an error (see 2.2). If nothing happens after you've waited, go to the Terminals Panel and see if there is any terminal corresponding to your shortcut. If yes, click on the Show / Hide button to make it appear. Most of the time, this is due to a problem in your *command line*.
There you will be able to see the terminal window, and then see where the problem is.

.

2.4 I'm under Windows connecting to the Runtime System using SSH. The terminal closes automatically; is this normal ?

Yes. fgltty uses the SSH protocol to send the command line. Your problem is due to the SSH protocol itself; if all the programs are closed, ssh will close the connection. That's why the terminal window is also closed. To avoid this, you can start a shell at the end of your command (add "bash;" for instance). Then, as the shell will be running, the connection will not be closed.

.

2.5 When starting a http shortcut, I got "Error GET : xxx" or "Error POST: xxx"; what does it mean ?

It means that there is a network problem between GJC and Genero Application Server. The message you are likely to see is "Error GET: Connection Refused", which means that either the computer hosting the Application Server is not reachable, or that the connection was rejected. Please check your shortcut configuration and the Genero Application Server configuration.

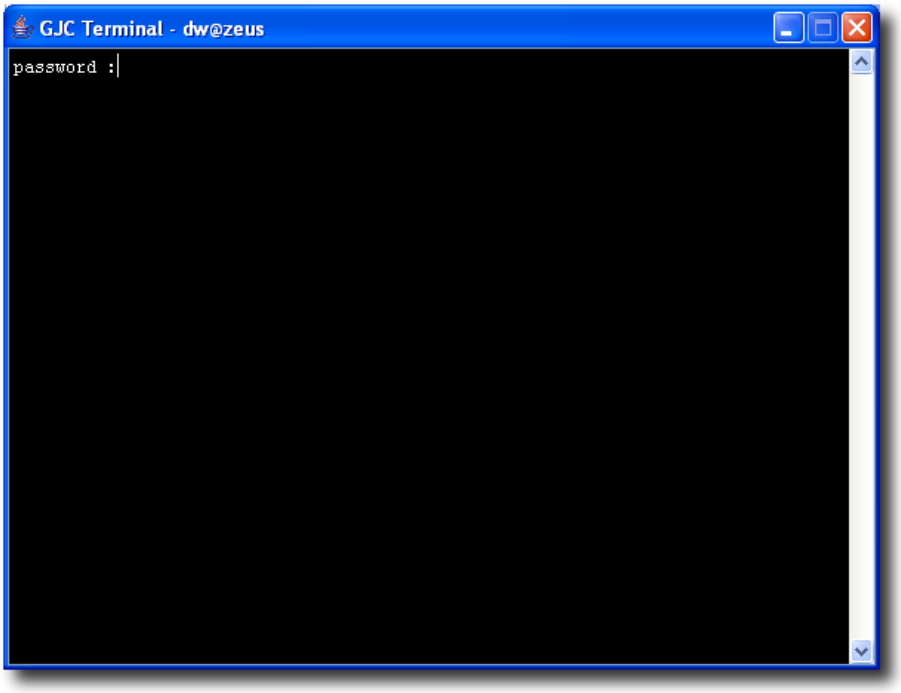
.

2.6 I've unchecked "This connection needs a password", but GJC still asks me for a password, why ?

First of all, you have to see who exactly is asking you for a password.

GJC asking for a password:

Terminal asking for a password:



When you unchecked "This connection needs a password", GJC will not ask you for a password and therefore will not transmit password to Terminal. But, if your authentication system is not correct, the host will still ask for a password. This is why Terminal asks you for the password.

To correct this, check your authentication system or contact your system administrator.

.

2.7 I have a lot of Terminals that stay in the background even though no application is running. Where is the problem ?

There is no way for GJC to know if the terminal that has been started is related to any application.

- You may start shortcut A, then B, and application B starts before application A.
- You may start a shortcut called A that starts an application called B.
- You may start a shortcut A that starts applications A, B.

So, GJC can't be sure that, if an application is closed, it should close the terminal.

To close the terminal, check the Terminals Panel: there are all your login sessions that can be Terminated by clicking on the corresponding button.

You should either:
1) add "; exit" to the end of your command string (eg.: @FGL ; cd $FGLDIR/demo ; fglrun ia.42r ; exit)
or 2) use exec to replace process (eg. @FGL ; cd $FGLDIR/demo ; exec fglrun ia.42r)

.

2.8 The buttons in the shortcut pannels are grayed, what should I do to modify my shortcuts ?

You are running GJC in "user" mode. To change your configuration options and you shortcuts, you have to star GJC in admin mode, using the right command line option.

.

2.9 What does the "Unexpected protocol version sent by the runtime system" message mean ?

The protocol used by the Runtime System and the GJC to communicate may change depending on new features added to Genero. The message "Unexpected protocol version sent by the runtime system" is displayed when the Runtime System and the Front End use a too different protocol to work perfectly. You can press "Continue", but this is likely to cause misbehaviours that may lead to GJCs crash.

We recommend to always use GJC with a Runtime System version designed to work together.

.

3. Network Problems

3.1 Can I change the listening port ?

Yes. -p (or --port) parameters allows you to change the port: `java -jar gjc.jar -p 3200` will make GJC listen on port 3200. In every case, if the wanted port is not available, GJC will try the next port and listen on the first free one.

.

3.2 I've started GJC on port 6400 and it seems it is listening on port 6401... Why ?

If the port you specify is not chosen by GJC, it means that it was not free. This may be due to:

- another instance of GJC which already listens on this port,
- an instance of another Four J's Front End which also listens on this port.

Note: If, during an application, a WINEXEC statement is performed, GJC will launch a local application. Until all these applications are closed, the listening port will be busy, even if you close GJC. So if you restart GJC while at least one application has not terminated, GJC will jump to the next port. The same situation appears with Terminals.

.

3.3 As I use SSH, my connection is completely secured, right ?

It depends on the way you've created your shortcut. If you've entered a "forwarded port", SSH tunneling will be set and then your connection is **secured**. If this value is not entered, SSH is only used to secure the connection when GJC needs to start an application on a distant host, and not the complete communication.

.

4. Logging System

4.1 Why a logging system ?

The logging system has two main utilities, debugging and demo making.

Debugging: if you notice any problem in the GJC (especially an unexpected crash), you can record the scenario that caused the problem and send it back to us. This will allow us to solve the problem.

Demo Making: you may want to make a demonstration for your application, but you don't want to export the installation of the Database and the Runtime System. Recording

a demo will allow you to create a scenario of your application and move it anywhere. The only requirement is GJC.

WARNING! Not everything is logged by GJC. See (4.2) for more details.

.

4.2 I have moved some columns on a table during a demo, but the replay does not change the order; is this normal ?

Unfortunately, yes. The logging system only logs the communication between the GJC and the Runtime System. Everything that is completely handled locally (like moving columns) will not be recorded.

---

5. Concerning Applets

5.1 What is the difference between ja/r and wa/r for HTTP connection ?

GJC can be used with the Genero Application server in two modes:

- as a module integrated into the Application Server (as an Applet),
- totally independently.

When you run as a module, it works as follows:

In your Internet Explorer browser, you enter an URL like
`http://server:port/path/wa/r/applicationName` where

- `server` is the server name,
- `port` is the port of the server,
- `path` is a certain path, depending on your configuration (could by empty, could be "cgi-bin/fglccgi")
- `applicationName` is the name of the application you want to start

Then the Application Server will send back an HTML page embedding GJC as an Applet, and it will start the application.

When you run independently, it works as follows:

In the shortcut system, you can specify either `server`, `port` and `applicationName`, or a complete url. When you specify each piece of information, it is equivalent to the following url:

```
http://server:port/ja/r/applicationName
```

Then an internal protocol is sent to the front end.

wa/r	What happens:	The Application Server will send an HTML page with GJC Applet embedded and start an application.
	When to use it:	This should be used in your **web browser** to start the GJC Applet and an application from the application server.
ja/r	What happens:	The Application Server will send an internal protocol understood by the GJC.
	When to use it:	This should be used in the **shortcut system** to specify a shortcut using the HTTP connection.

.

---

6. GJC's behavior

6.1 GJC is blocked when a system dialog is open: is this normal ?

System dialogs depends on the... Operating System. When the dialog is open, GJC waits for an answer of the Operating System. Everything which is sent by the Runtime System is stored and processed once the dialog is closed by the user.

So this is normal that if you have for instance the "printer" dialog open (after a `START REPORT` with `EXPORT DBPRINT=FLGSERVER`), GJC seems to be blocked and do not start or carry on any other application. Close the dialog and it will carry on working as expected.

Glossary

This documentation uses several terms that must be clarified for a good understanding. Here is a short description for all these terms:

Product

The Product defines all software components that compose the information system managing a given domain. Usually, the domains covered by programs written in BDL are business oriented.

End User

The End User is the person that uses the Product; that person works on hardware called the Workstation.

Programs

The Programs are the software components that are developed and distributed by the supplier of the Product. Programs typically implement business rules and processing, usually called Business Logic. Programs are executed by the Runtime System on the Application Server machine. These components are typically p-code modules, forms and additional files.

Developer

The Developer is the person in charge of the conception and implementation of the Product components.

Application Data

Application Data defines the data manipulated by the Product. It is typically managed by one or more Database Systems. The Application Data has a volatile state when loaded in the Runtime System, and it has a static state when stored in the Database System.

Database

The Database is a logical entity regrouping the Application Data. It is managed by the Database System.

Database System

The Database System is the software that manages data storage and fetching; it is usually installed on the Database Server machine and is supported by a tier software vendor. It is the software managing the Data in the Three-Tier C/S model.

Development Database

The Development Database is the Database used in the application development environment.

Production Database

The Production Database is the Database used on production sites.

Front End

The Front End is the software that manages the display of the User Interface on the Workstation machine. This component is historically called "The Client", in a thin Client/Server context. It is the software managing the Presentation in the Three-Tier C/S model.

Runtime System

The Runtime System is the software that manages the execution of the Programs, where the Business Logic is processed. It is typically implemented by the Dynamic Virtual Machine (DVM) and historically called "The Runner". It is the software managing the Processing in the Three-Tier C/S model.

User Interface

The User Interface defines the parts of the Programs that interact with the end user, including interactive elements like windows, screens, input fields, buttons and menus. It is displayed on the Workstation. This can typically be implemented by different kinds of Front Ends, based on ASCII terminals, graphical platforms (MS Windows, X11) or even on web basements like HTML over HTTP.

Workstation

The Workstation identifies the hardware used by the End User to interact with the Product. It can be an ASCII Terminal, a PC, a diskless station or even a cellular phone, as long as a Front End is available on that hardware.

Security Terms

The security section of the documentation uses several terms that must be clarified for a good understanding. Here is a short description for all these terms:

Firewall Router

This is a device that isolates the corporate network from the Internet. It typically allows connections to the Internet, but also prevents connections from entering. They can usually be configured to allow/prevent several conditions. They can be configured to allow a port connection from the Internet to go through to a machine. This can be done either by allowing the connection straight through or translating it to a different port.

NAT

Network Address Translation is a method of allowing computers to access the Internet without having them be assigned real Internet addresses. The connections must originate from the internal machines to reach Internet addresses. The NAT router will then put these on the Internet using the router's IP address. When data is returned it forwards the data to the requesting internal machine. Part of this process includes mapping what internal IP/Port combinations correspond to external port usage. Doing so allows the router to know where data needs to be sent when it returns. Special port mappings can be made to specific internal IP addresses to support connections originating from the Internet. Other configurable values might be session timers that will be explored in the section.

Private Network

This is the network used in the corporation that is private and trusted. Most companies tightly control what is plugged in so they can ensure the data is safe.

VPN

Virtual Private Network is a method of tunnelling through an existing connection back to the corporate LAN. It provides end-to-end encrypted connections. These types of connections are usually equivalent to being plugged into the office LAN.

Encryption of all Data

Genero requires a TCP connection for the GUI data transmission. If the GDC short cuts are being used there is also a connection needed to start the application that may require a log in. Both connections in this case are encrypted.

Password / Login Encrypted

Genero logs in and executes an application when the short cuts are used. This connection is encrypted. The connection carrying the GUI data is not encrypted.

Keep Alive

Typical TCP connections don't cause any network traffic when idle unless the KeepAlive flag is set. This flag will prevent the session from timing out and thus prevent the session from closing. This also assumes that the firewalls don't expire the session during the keep alive interval.

Port Forwarding

The method referred to is implemented in the Secure Shell (ssh). The ssh can be told to listen to a port and tunnel it through an existing ssh session and present it to a port on the other machine. This method is used to listen to a port on the server side and direct the data to the GDC on the client side.

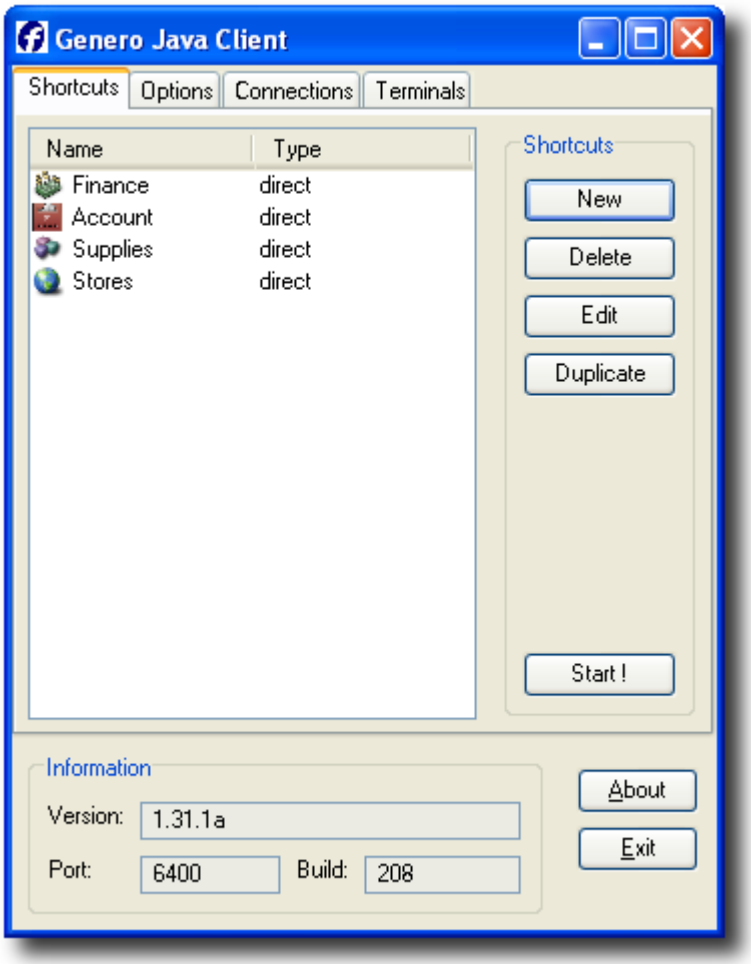
Shortcut System

Summary:

- Shortcut System Overview
- Administration mode
- Manage shortcuts
- Starting a shortcut
- Local Shortcuts

Shortcut System Overview

Genero Java Client (GJC) is able to store information allowing it to easily start applications. This information is stored as **shortcuts**, stored internally in the same way on each platform.



Administration Mode

By default, GJC starts in **user mode**. When in user mode, shortcuts and options cannot be modified.

To create shortcuts or modify a shortcut's options, you must start GJC in admin mode, using the "`--admin`" or "`-a`" command line option.

Manage Shortcuts

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

- The Runtime System is on a distant host and GJC starts it via SSH2 (Direct Connection),
- The Runtime System is on the same host and GJC starts it as a local application (Local Connection),
- The Runtime System is on a distant host, and GJC connects to it via Genero Application Server (Connection via AS).

The Shortcut Management System allows you to add a new shortcut, to edit, duplicate or remove an existing one.

Creating (or editing) a shortcut uses a Wizard to guide you as you create (or edit) your shortcut.

Starting a Shortcut

You can start a shortcut by:

- Double-clicking on the shortcut icon
- Selecting the shortcut from the list and clicking the **Start !** button.

Local Shortcuts

You can also create **local** shortcuts, which are shortcuts stored locally for each user. This feature is useful if you share GJC on a network drive and don't want the user to modify the common shortcuts.

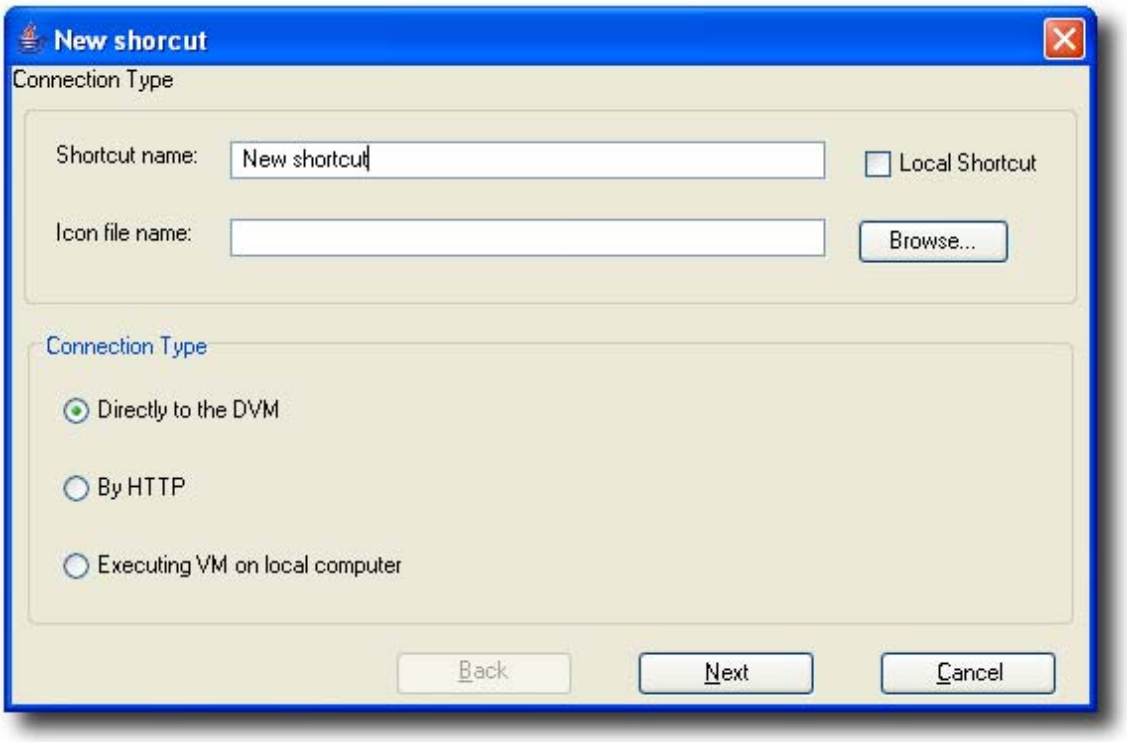
By default, your shortcuts are saved in the **$GJCDIR/etc/config.xml** file. When the **config.xml** file is read-only, any modification to a non-local shortcut displays a warning and creates a local copy of the shortcut.

Shortcut Wizard

Summary:

- Wizard Start
- Direct Connection Shortcuts
- SSH Tunneling
- Local Connection Shortcuts
- Connections via Application Server
- Shortcuts and Environment Variables

Start the Wizard



When you start the Wizard, you are asked for the following:

- A name for the shortcut.
- A file name to be used to display an icon associated with the shortcut. This field is optional; if you do not specify the name of an icon file, the default icon for your GJC installation is used.
- The Connection Type for your shortcut:
    - The Runtime System is on a distant host, and GJC will start it via telnet, rlogin or SSH (Direct Connection), select "Directly to the DVM".

- The Runtime System is on the same host, and GJC will start it as a local application (Local Connection), select "Executing VM on local computer".
- The Runtime System is on a distant host, and GJC will be connect to it via Genero Application Server (Connection via AS), select "By HTTP".

Depending on the Connection Type selected, the Wizard displays the next appropriate screen.

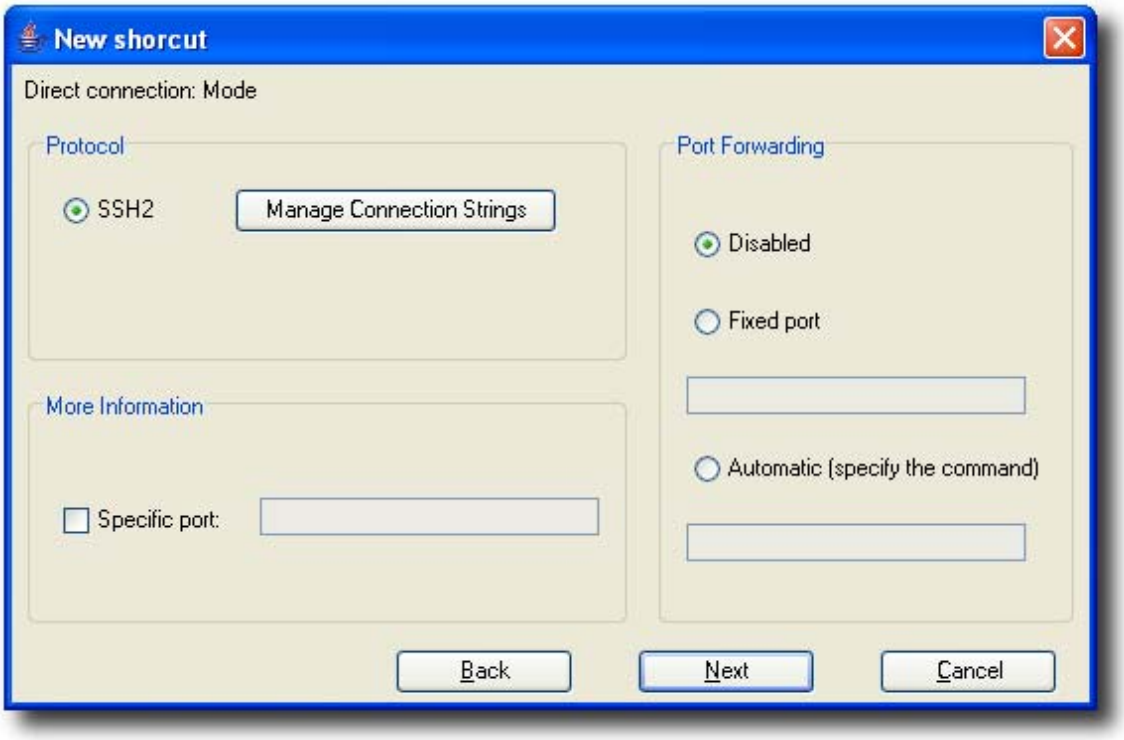
Direct Connection Shortcuts

Mode screen:

In Direct Connection mode, the Runtime System is directly connected to the GJC using a TCP/IP network. To start your program on the Runtime System host, GJC connects to the host using SSH2. You can specify an alternative port, if your configuration needs it.

WARNING! This connection mode is only used when GJC connects to the host to start the application. Currently, the communication between the Runtime System and GJC does not use SSH2. Tunneling can be used to secure your connection.

.



There you can specify **connection strings**:

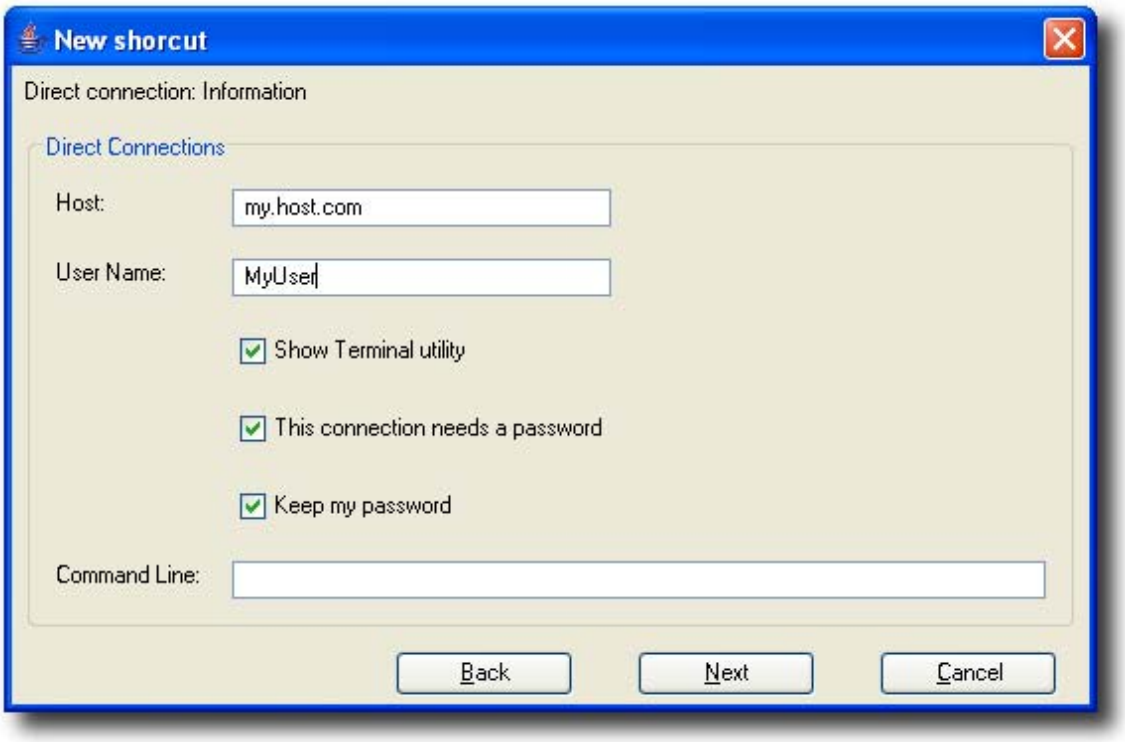
This table is used to tell GJC what to do when the Runtime System host displays a given string on the terminal. GJC can perform the following actions:

- Ask the user for a value, and send it back.
- Display a message to the user.
- Ask for a password.
- Send the shortcut password.
- Send the shortcut command.
- Return a defined string.
- Ignore the Runtime System string.
- Get a free port number for Port Forwarding.

"Order" specifies in which order GJC tries to recognize the different strings. You can specify for each string whether it should be recognized only once or every time.

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

Information:



In this case, you have to provide the following information:

- the hostname where the Runtime System is hosted. This can be omitted if you use the -host command line.

- the username you are using to connect to the host. This can be omitted if you use the -user command line.
- the command line that will be executed to start the application on the Runtime System side.

If ***This connection needs a password*** is checked, GJC 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.
WARNING! GJC will not modify your configuration to allow you to connect without a password. It is up to you or your administrator to manage this.
If ***Keep my password*** is checked, GJC keeps in memory the password you enter the first time you start a shortcut, and reuses it when you restart. The password is kept in memory, and is lost if you stop GJC.
WARNING! GJC never stores your password in a file or elsewhere. The password is kept in memory while GJC is launched, and is forgotten once GJC is stopped.
If ***Show Terminal Utility*** is checked, the window of GJC Terminal, our Emulation Terminal Utility, will be visible. (Please refer to Terminals Section). This could help you to check if your command line is valid or not.

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

Tag

	Replaced by
@FGL	FGLSERVER=*<IP Address>:<serv num>*

You can use one of the @FGL variants depending on your system:

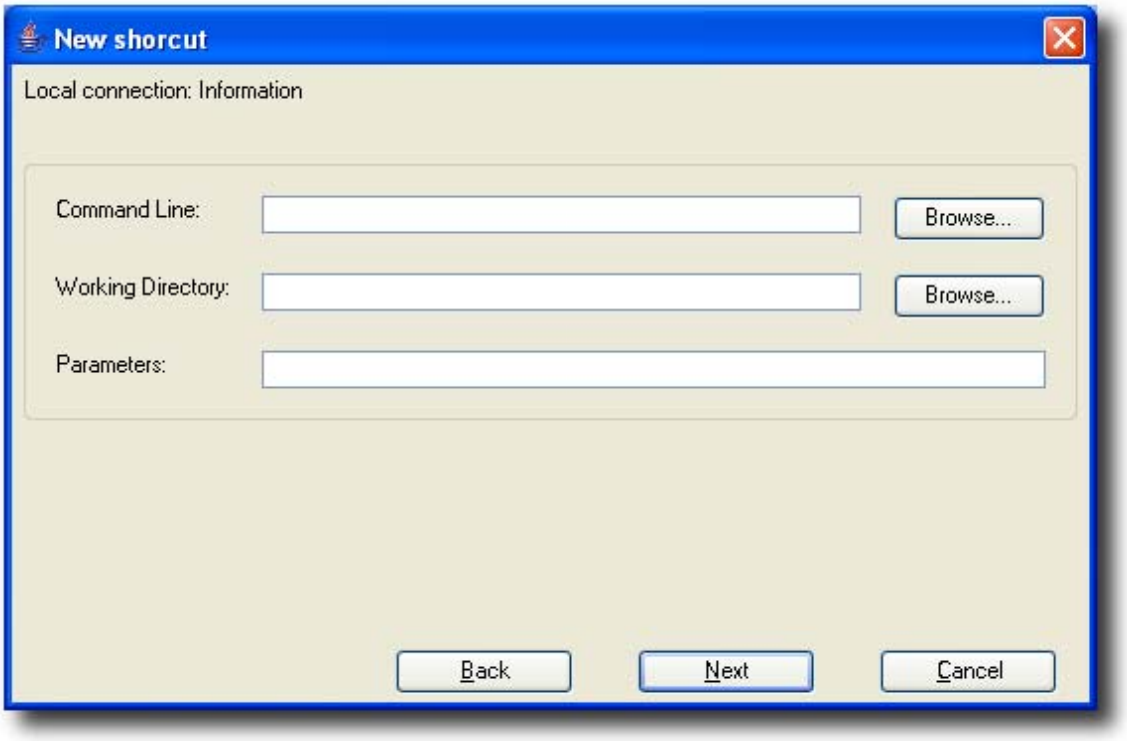
@FGLNT	set FGLSERVER=*<IP Address>:<serv num>*&&set FGLGUI=1
@FGLCSH	setenv FGLSERVER "*<IP Address>:<serv num>*";setenv FGLGUI 1
@FGLKSH	FGLSERVER="*<IP Address>:<serv num>*";export FGLSERVER;FGLGUI=1;export FGLGUI
@SRVNUM	*<GJC listening port - 6400 (The second part of FGLSERVER)>*
@PORT	*<GJC listening port>*
@USR	*<Client current user name>*
@USER	*<User name on the remote system>*
@IP	*<IP address of the client computer>*
@COMPUTER	*<Machine host name>*
@E_SRV	export FGLSERVER
@4GLSRVVER	*<GJC version>*

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

SSH2 Tunneling

For information on SSH2 tunneling, refer to the GJC and SSH2 section.

Local Connection Shortcuts



Information

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

- `fglrun` started in the application directory or
- a batch file that contains `fglrun` instructions for all applications.

You must provide the following information:

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

You can have, for instance:

Command Line	Working Directory	Parameters	Remarks
`fglrun`	`/home/fgl/demo/`	`stores.42m`	`fglrun` should be in the PATH.
`c:\fourjs\fgl\bin\fglrun.exe`	`c:\genero\demo`	`stores.42m`	
`c:\demos\stores.bat`			The **stores.bat** file is a batch file that sets the environment and starts the program.
`C:\WINNT\system32\CMD.EXE /K "d:\fjs\fgl\envcomp.bat && fglrun D:\app\gift.42r"`	`D:\app`		This starts the **envcomp.bat** script in the **$FGLDIR** directory to set the environment, then starts the gift application. The working directory is the directory where `fglrun` can find the required files.

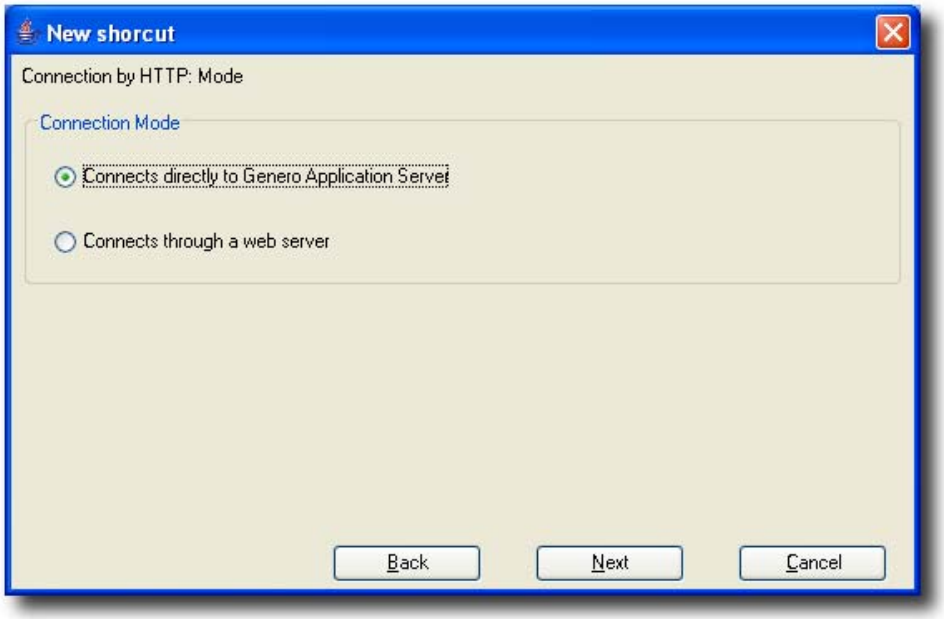
WARNING ! If you have the following configuration:

```
c:\fourjs\fgl\bin\fglrun.exe c:\genero\demo stores.42m
```

you must be sure that all the environment variables are set **before** starting the application. This can be done in the "Environment Variables" system dialog.

Connections via Genero Application Server

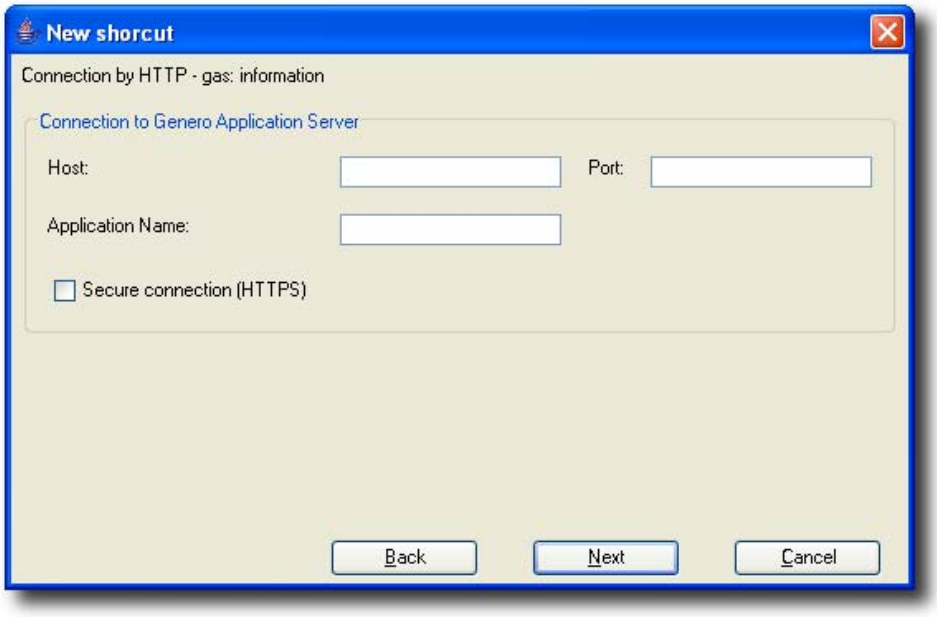
In this mode, the GJC connects to the Runtime System via the Genero Application Server using the HTTP protocol.



GJC connects in one of two ways:

1. Directly to a Genero Applications Server that runs as Web Server.

If this method of connecting is selected, the following information is required:

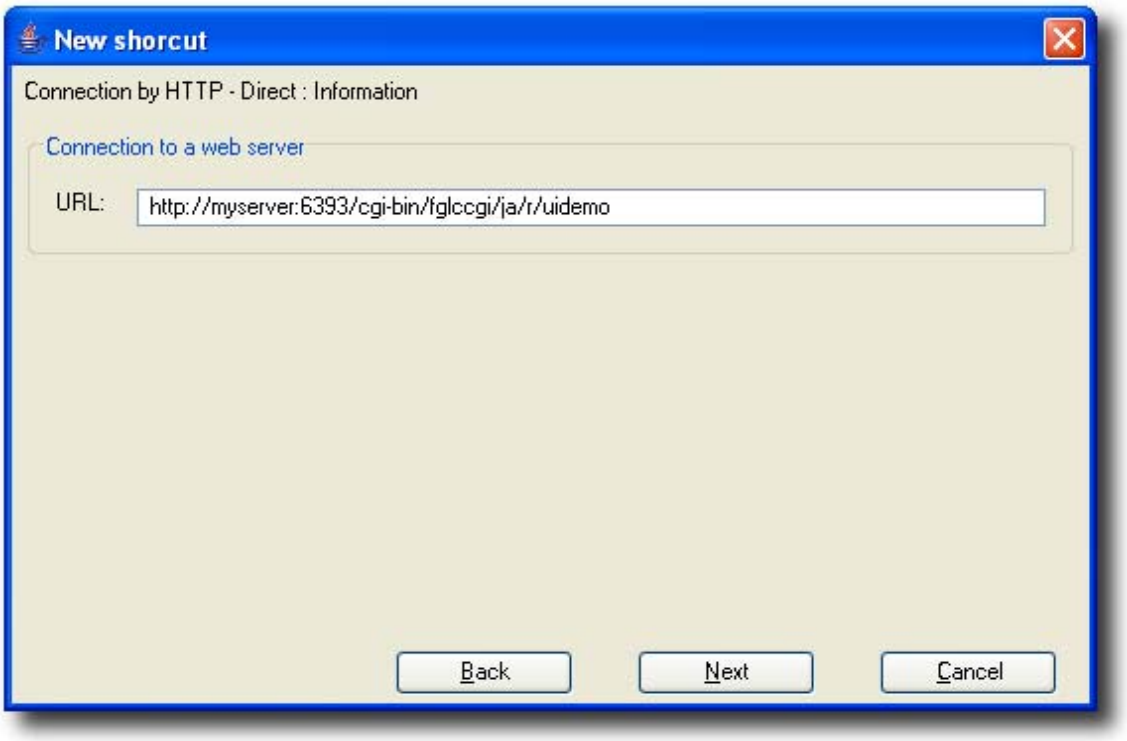


- The host where Genero Application Server is running.
- The port where the Genero Application Server is listening for a connection request.
- The name of the application.

For more information on configuring applications, see the *Genero Application Server* documentation.

2. Via a Web Server.

The GJC can connect to the Genero Application Server via a Web Server. In the following example, the connection is made using CGI:



When connecting to the Genero Application Server via a Web Server, only the URL is required. With our CGI example, the URL looks like: `http://myserver:6393/cgi-bin/fglccgi/ja/r/uidemo.`

For more information about URLs, see the Frequently Asked Questions section. For more information on configuring applications, see the *Genero Application Server* documentation.

Shortcuts and environment variables

In the following fields, GJC will replace any `$xxx` (X11 / Mac OsX) or `%xxx%` (Windows) with the corresponding environment variables:.

direct	`host`, `username`, `commandline`
http	`url`
local	`command line`, `working directory`, `parameters`

Connections Panel

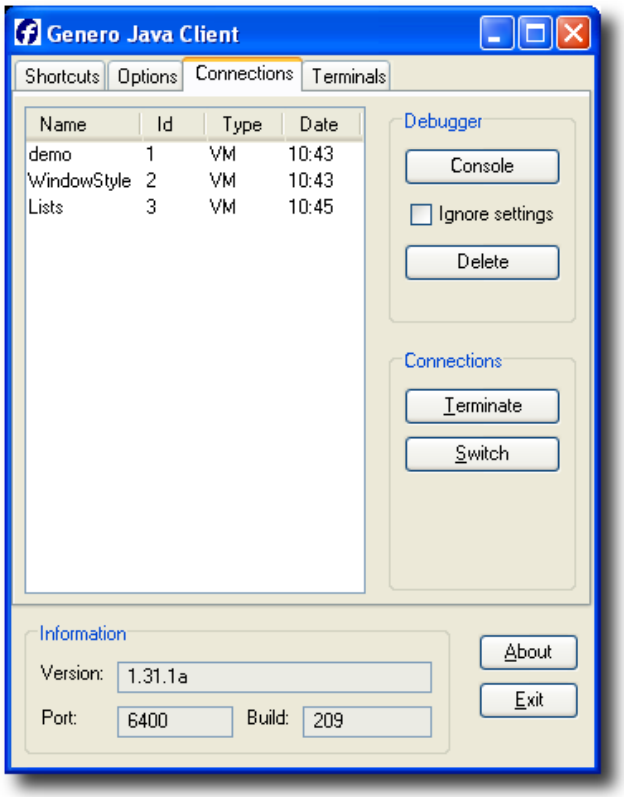
Summary:

- View the application list
- Find a specific application
- Stop an application
- Use the Debug Console

View the Application List

The **Connections Panel** lists all applications handled by the GJC. For each application, the following information displays:

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



Find a specific application

The Switch feature allows you to find your application easily if a lot of programs are launched. To find a specific application and bring that application to the forefront, from the Connection Panel:

1. Select the application from the application list.
2. Click the "Switch" button.

The selected application is brought to the forefront, and the focus is set on the current window.

Stop an application

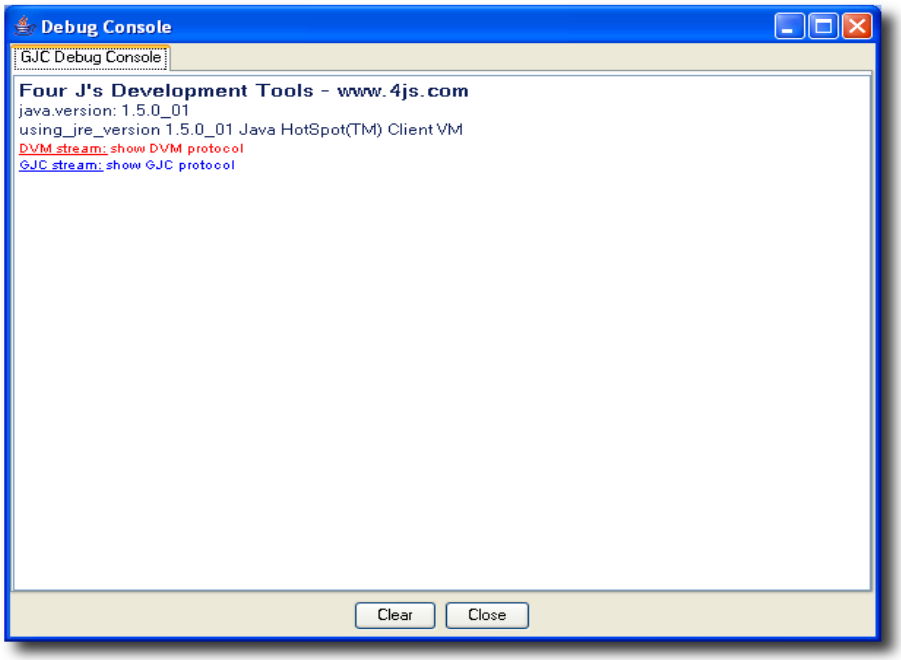
If you want to stop an application, from the Connection Panel:

1. Select the application from the application list
2. Click the "Terminate" button.

This sends the information to the Runtime System; the application is stopped by the GJC.

Debug Console

The debug console shows the communication between the GJC and the Runtime System.



To use the Debug Console:

1. Click the "Console" button.

The Debug Console window appears. In this window, debug information displays:

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

WARNING! The debug console is only available in debug mode.

Terminals Panel

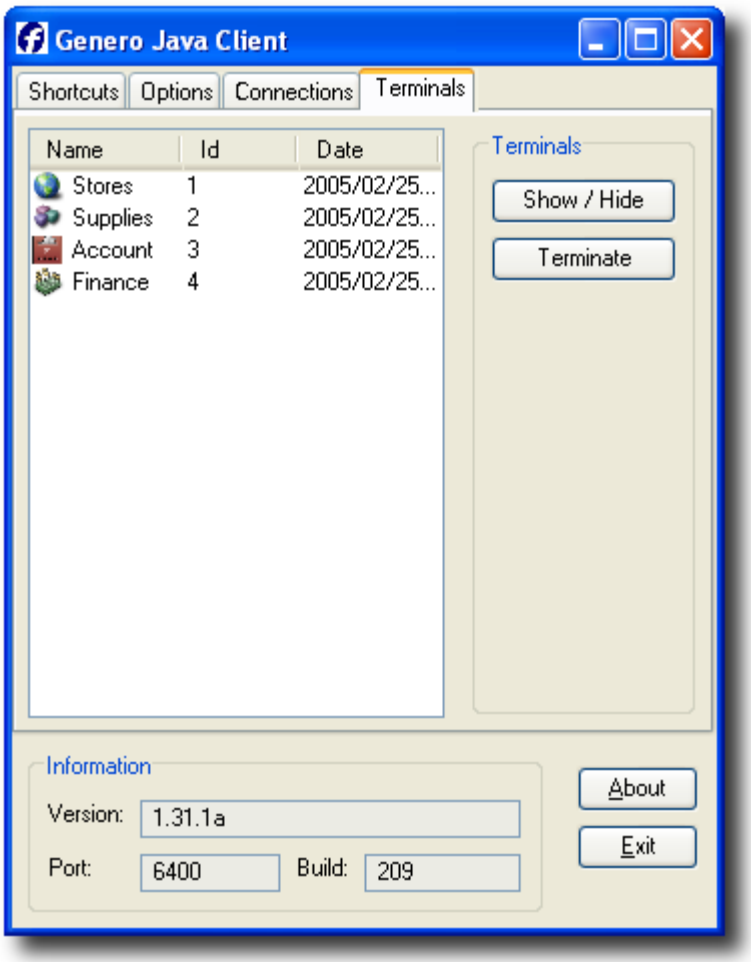
Summary:

- Overview
- Show or Hide a GJC Terminal
- Terminate (stop) a GJC Terminal

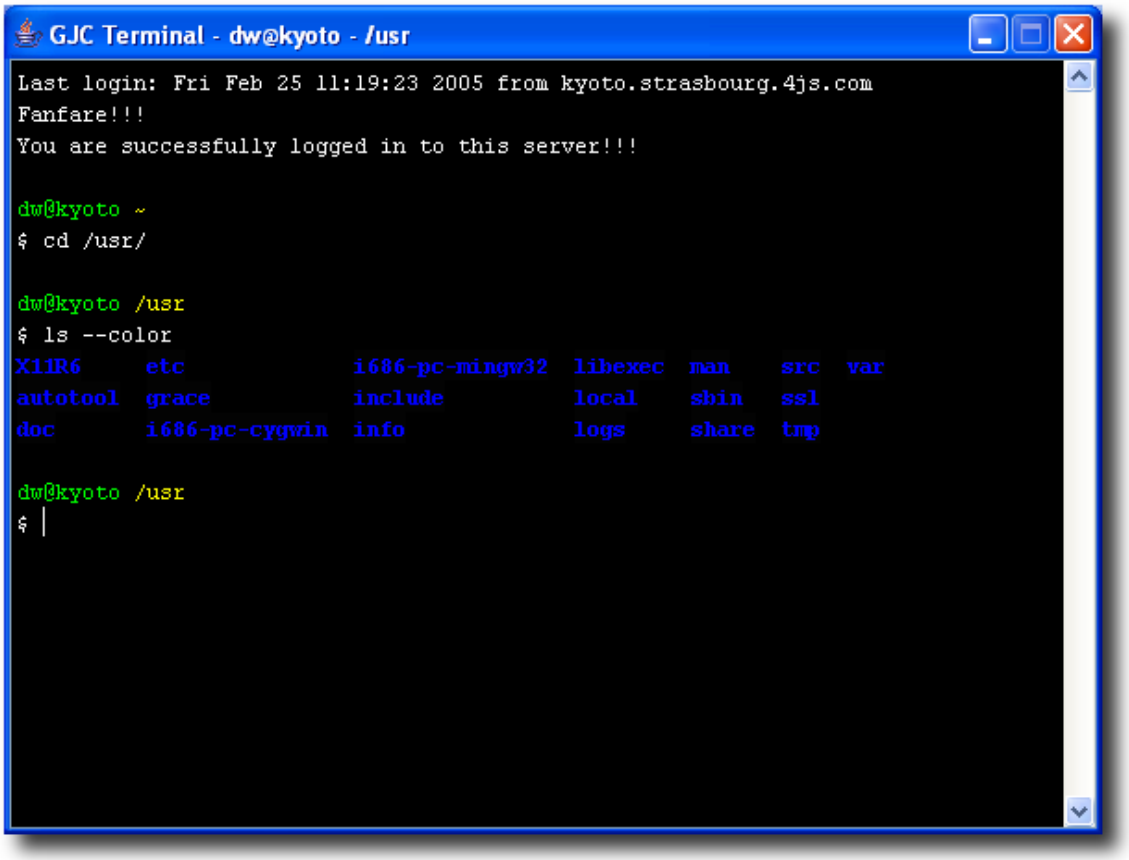
Overview

Shortcuts are using a terminal emulation utility to connect to the system hosting the Runtime System. Each entry in the list displayed by 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 GJC Terminal.



Show or Hide a GJC Terminal

The **Show / Hide** 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 or not. With this button, you are able to show a hidden terminal, or hide a visible one.

This button is typically used to determine why your application has not started. Showing the Terminal Utility displays what has happened.

Terminate (Stop) a GJC Terminal

The **Terminate** button allows you to terminate the selected Terminal Utility.

Stored Settings

If you change some graphical aspects of your application, GJC will store the changes and reload them when the same application is run again. This section lists all parameters that are stored.

- Window Settings
- Table Settings
- TextEdit Settings

Window Settings

Global Settings

These settings are stored under
.**Application**\<***applicationName***>**Windows**\<***windowName***>
<*applicationName*> is taken from the "text"-Attribute of the `UserInterface` Node
The initial position of the Window is stored with the "posX" and "posY" Key, the initial state (maximized or normal) under the "state" key.

posX - x position of the upper left corner of the top-level window

posY - y position of the upper left corner of the top-level window

state - "normal" or "maximized"

Remarks: The window position and initial state of the settings are only taken into account if the "position" style attribute of the window is "default" (not "center" and not "field").
If an application is popping up the first time and no settings are available, it is positioned automatically by the Window-Manager/GJC.

Start Menu settings

The **width** of the **StartMenu** (when the style is "tree").

Table Settings

The first time the table is sent, the GJC stores the initial settings prefixed by DVM_ : **DVM_numColumns**, **DVM_pageSize**, **DVM_sortColumn**, **DVM_sortType**.
Each time the Table is sent again, GJC compares the initial settings with the stored settings. If something has changed, it means that the 4GL program has been changed, and then the user stored settings for this table are reset.

Global Settings

These settings are stored under **.\Forms\Table Columns\<*FormName*>.<*ArrayName*>**
<*FormName*> is the "name" attribute of the "Form"-Node in the UI-Tree
<*ArrayName*> is the "tabName" attribute of the "Table"-Node.
This allows the settings to be re-used in multiple applications.

numColumns - the number of columns the table had when this setting was written; if a table with a different number of columns is created under the same settings identifier, these settings will be ignored.

pageSize - the number of rows the Table had when the Table was closed, used to restore exactly this number of rows if the table is shown again.

pageSizeInitial - the number of rows the Table had when the Table was created; if a table with the same identifier is created with a different number of rows the "pageSize" settings key is ignored.

pixelWidth - the pixelWidth the table had when it was closed the last time, will be restored the next time it is shown.

sortColumn - the sort column the table had when it was closed; will be restored if the table is shown again. A Reset of this saved column (to -1) is only possible when clicking on the Right Mouse Click-Context Menu at the Table header and choosing "Reset To Defaults".

sortType - the sort Type "asc" or "desc" the table had when it was closed and a sortColumn was saved; will be restored if the table is shown again.

Column Settings

Individual column settings are stored under **.\Forms\Table Columns\<*FormName*>.<ArrayName>\<*columnNumber*>**

pixelWidth - the pixelWidth this column had if the table was closed, will be restored the next time the table is shown.

realColumn - the actual visible number (order) of the column; if the user had moved columns by drag and drop, this number is different from <columnNumber> and will be restored the next time the table is

shown.

text - "text" attribute of the "TableColumn" Node, used for debugging.

visible - if a user had used the context menu of the table header to switch off/on a column, the result of this operation is saved under this key and will be restored the next time the table is shown.

TextEdit Settings

Global Settings

These settings are stored under
.\Forms\TextEdits\<*FormName*>.<*TextEdit*>
<*FormName*> is the "name" attribute of the "Form"-Node in the UI-Tree
<*TextEditName*> is the "colName" attribute of the "Textedit"-Node.
This allows the settings to be re-used in multiple applications.

height - the height this textedit had if the form was closed, will be restored the next time the form is shown.

width - the width this texedit had if the form was closed, will be restored the next time the form is shown.

Command Line

Summary:

- Command line options list
- Warnings
- Examples

Command line options list

GJC handles the following options:

Information		
`-h`		This will make GJC display the About Box.
Network, System		
`-p` `--port`	`new_port`	GJC will listen on the *`new_port`* port (if available).
`-n` `--new`		Starts a new instance of GJC.
`-D`		Starts GJC in debug Mode (debug Tree and debug Console are active)
`-A` `--Authentication`	`security level`	Sets GJC's security level regarding Runtime System's connection.
Start Application		
`-S` `--Start`	`shortcut_name`	If GJC is still not launched, GJC will start, and then, the shortcut named `shortcut_name` will be started.
`-s` `--startDirect`		If GJC is still not launched, GJC will start, using the informations given by -U, -H, -T, -P and -C to connect to a DVM.
`-U` `--User`	`user name used`	The specified `user` name will be used when starting a Direct Connection. This can be used if you share GJC, then each user can create a link to the bin and differentiate the shortcut that will be launched.
`-H` `--Host`	`host name`	The specified `host` name will be used when starting a Direct Connection, either with a defined shortcut (with -S), either directly (with -s).
`-P` `--Password`	`password`	The specified `password` will be used when starting a Direct Connection, either with a defined shortcut

		(with -S), either directly (with -s).
-C	command_line	The specified command_line will be used when starting a Direct Connection, either with a defined shortcut (with -S), either directly (with -s).
--Cmd		
-T	connexion_type	When starting an application with -s, defines which protocol should be used. Values can be: TELNET, RLOGIN, SSH, SSH2. Default is SSH.
--Type		
Start GJC		
-a		Starts GJC in admin mode.
--admin		
-M		Starts GJC minimized.
--Minimized		
-X		Closes GJC if there is no more application or terminal running.
--close		

Warnings

- -S and -s must be used separately. -S is used to start and existing shortcut, and -s to start an application using the command line.
- When using -s, you must specify at the host and the command line. You will be prompted for a username and password if needed.

Examples

`java -jar gjc.jar -p 6350` starts GJC on port 6350.

`java -jar gjc.jar -S demo` starts GJC, and the shortcut named "`demo`"

`java -jar gjc.jar -S demo -U smith` starts GJC, and the shortcut named "`demo`" using "`smith`" as user name.

`java -jar gjc.jar -s -T SSH2 -U smith -H server -P whatisthematrix -C "cd demo ; fglrun demo" -X` starts GJC, then connects to "`server`" as "`smith`" with the password "`whatisthematrix`". once connected, performs "`cd demo ; fglrun demo`", and closes the GJC when all the applications or terminals are over.

Local Actions

Summary:

- Overview
- List
- Interrupt Local Action

Overview

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

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

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

Example

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

In this example, when this button is clicked, the focus goes to the next field.

Example

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

In this example, the Return key is an accelerator to go to the next field. When the Return key is pressed, the focus goes to the next field.

List of Local Actions

Edition		Default Shortcut
`editcopy`	Copies the selected text into the clipboard.	`CTRL-C`
`editcut`	Cuts the selected text into the clipboard.	`CTRL-X`
`editpaste`	Pastes the content of the clipboard into the current field.	`CTRL-V`

Navigation in fields		Default Shortcut
`nextfield`	Goes to the next field.	`TAB`
`prevfield`	Goes to the previous field.	`SHIFT-TAB`
Navigation in Tables		**Default Shortcut**
`firstrow`	Goes to the first row.	`HOME`
`prevpage`	Goes back one page. (With TABLE, it follows Internet Explorer behavior).	`PRIOR`
`prevrow`	Goes to the previous row.	`KEY-UP`
`nextrow`	Goes to the next row.	`KEY-DOWN`
`nextpage`	Goes forward one page. (With TABLE, it follows Internet Explorer behavior).	`NEXT`
`lastrow`	Goes to the last row.	`END`
Interrupt		**Default Shortcut**
`interrupt`	Sends an interruption request to the Runtime System	

Interrupt

The interrupt action is enabled when the Runtime System is running without sending information to the front end. It allows the front end to send an interruption request, using a special communication system, named Out Of Band (OOB). When the Runtime System receives OOB, it sets INT_FLAG to 1. For more information on interruption requests, refer to 'The Dynamic User Interface' section in the Runtime System Documentation.

GJC Applet Overview

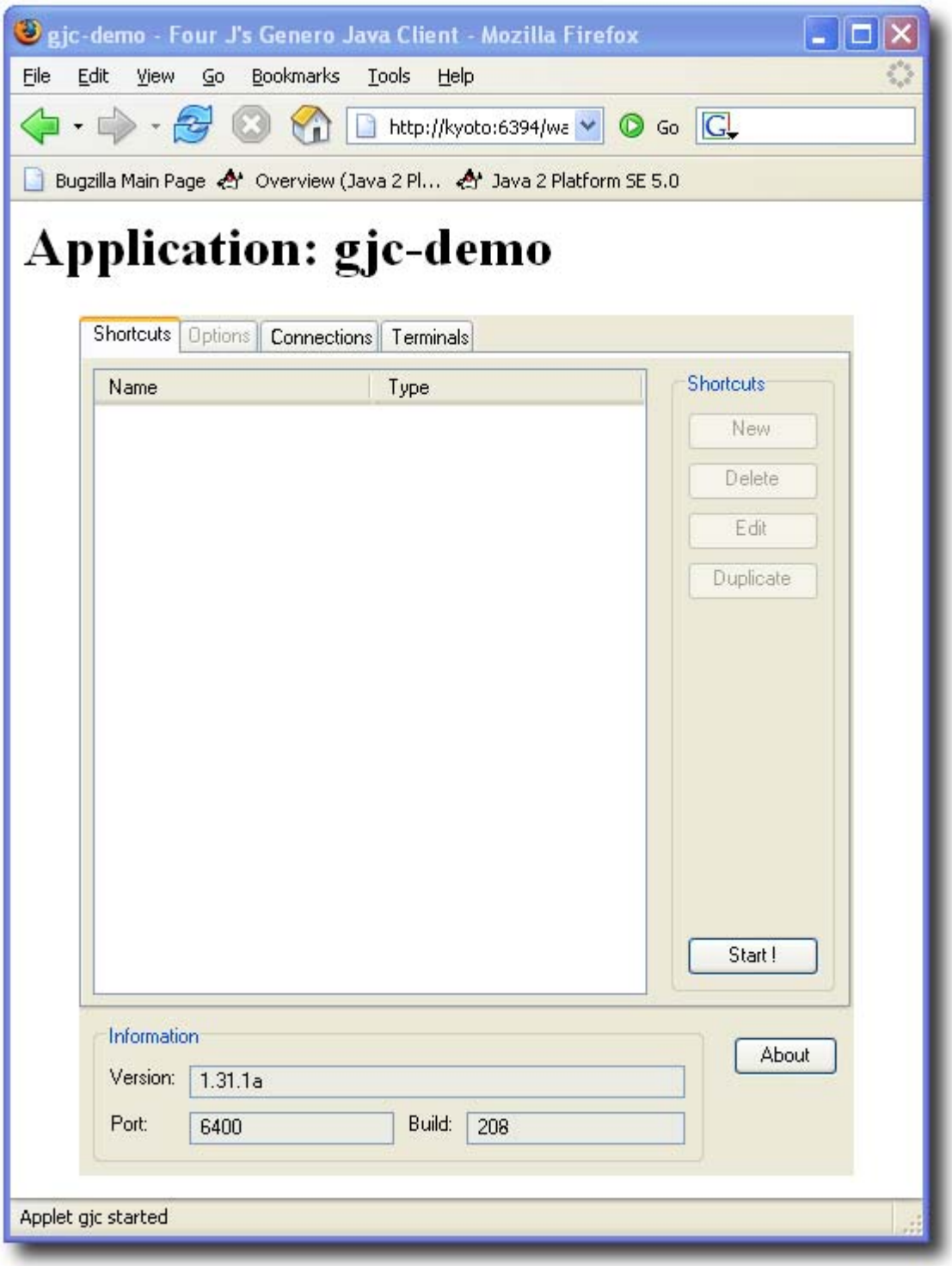
Summary:

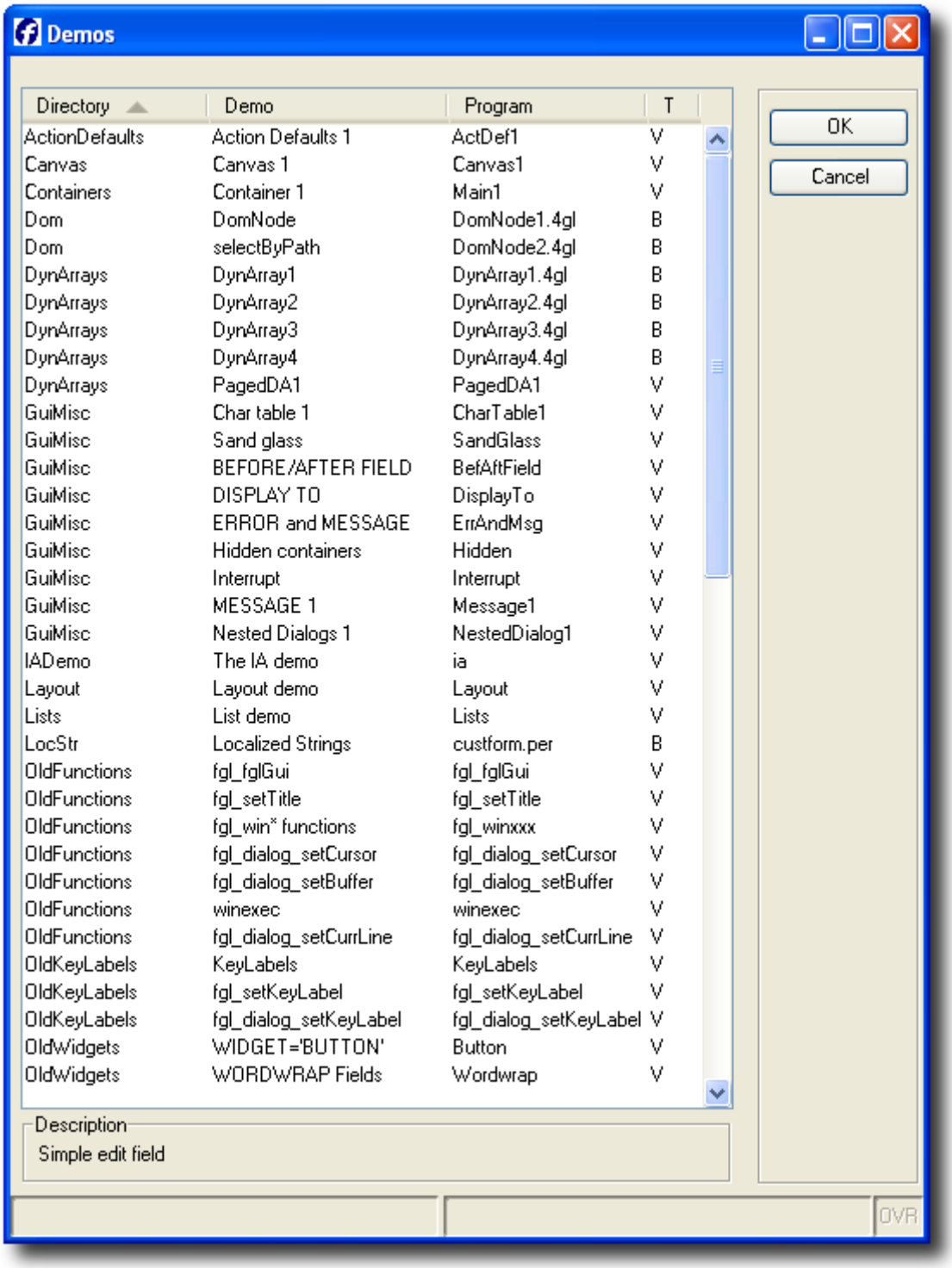
- Overview
- Installation
- Uninstallation

See also : Applet and GAS

Overview

Genero Java Client is also a Java applet. Packed into its **.jar** file, it can be placed into a Web site and then used directly with any browser that support Java Plug-in technology. Once a browser connects to the given URL, GJC displays within a Web page and can be used in the same manner as the "classic" version.


gjc-demo - Four J's Genero Java Client - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://kyoto:6394/wa
Go
Bugzilla Main Page
Overview (Java 2 Pl...
Java 2 Platform SE 5.0
Application: gjc-demo
Shortcuts
Options
Connections
Terminals
Name
Type
Shortcuts
New
Delete
Edit
Duplicate
Start !
Information
Version:
1.31.1a
Port:
6400
Build:
208
About
Applet gjc started

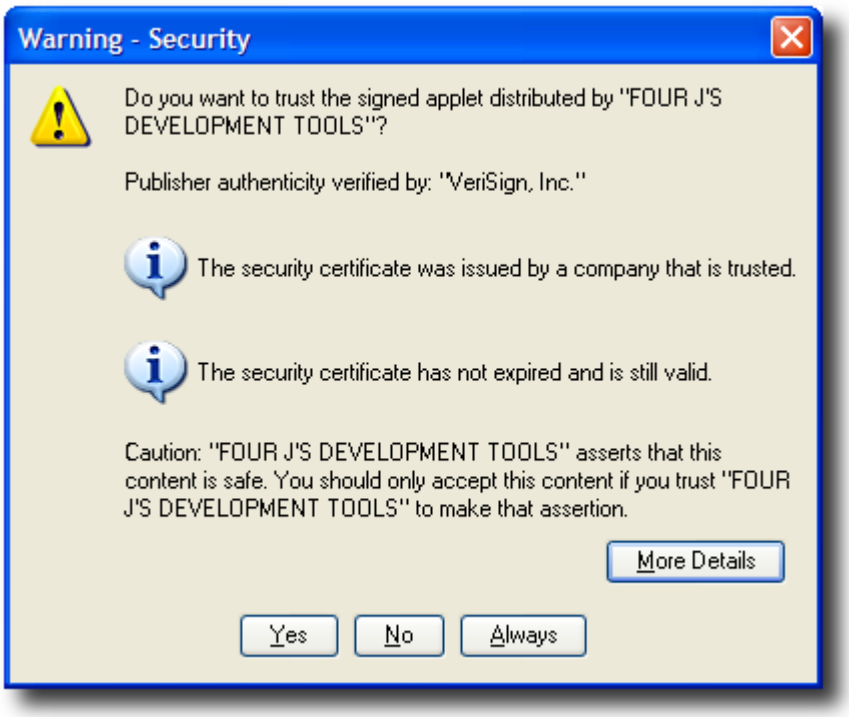

Demos
Directory
Demo
Program
T
ActionDefaults Action Defaults 1 ActDef1 V
Canvas Canvas 1 Canvas1 V
Containers Container 1 Main1 V
Dom DomNode DomNode1.4gl B
Dom selectByPath DomNode2.4gl B
DynArrays DynArray1 DynArray1.4gl B
DynArrays DynArray2 DynArray2.4gl B
DynArrays DynArray3 DynArray3.4gl B
DynArrays DynArray4 DynArray4.4gl B
DynArrays PagedDA1 PagedDA1 V
GuiMisc Char table 1 CharTable1 V
GuiMisc Sand glass SandGlass V
GuiMisc BEFORE/AFTER FIELD BefAftField V
GuiMisc DISPLAY TO DisplayTo V
GuiMisc ERROR and MESSAGE ErrAndMsg V
GuiMisc Hidden containers Hidden V
GuiMisc Interrupt Interrupt V
GuiMisc MESSAGE 1 Message1 V
GuiMisc Nested Dialogs 1 NestedDialog1 V
IADemo The IA demo ia V
Layout Layout demo Layout V
Lists List demo Lists V
LocStr Localized Strings custform.per B
OldFunctions fgl_fglGui fgl_fglGui V
OldFunctions fgl_setTitle fgl_setTitle V
OldFunctions fgl_win* functions fgl_winxxx V
OldFunctions fgl_dialog_setCursor fgl_dialog_setCursor V
OldFunctions fgl_dialog_setBuffer fgl_dialog_setBuffer V
OldFunctions winexec winexec V
OldFunctions fgl_dialog_setCurrLine fgl_dialog_setCurrLine V
OldKeyLabels KeyLabels KeyLabels V
OldKeyLabels fgl_setKeyLabel fgl_setKeyLabel V
OldKeyLabels fgl_dialog_setKeyLabel fgl_dialog_setKeyLabel V
OldWidgets \WIDGET='BUTTON' Button V
OldWidgets \WORDWRAP Fields Wordwrap V
OK
Cancel
Description
Simple edit field
OVR

Installation

Warning! If you are running Windows XP Service Pack 2, the installation is described in the GJC and Windows XP Service Pack 2 section of this manual.

As an applet, GJC cannot really be installed. However, the jar file containing the GJC is downloaded only when needed: the first time you connect to the Application Server where the GJC applet is installed, or when the applet is updated on the server). The jar file is stored in your browser cache and/or in the JRE cache system. Each time you clear it, you must download the applet again.

The first time you start the GJC as an applet, you must agree to trust the signed applet. When the security certificate warning window displays, click the **Yes** button to continue starting the GJC applet.



Uninstallation

As a Java applet, there is no uninstallation procedure. If you clear your browser and Java plug-in cache, the GJC applet is removed.

Applet and Application Server

Genero Java Client (GJC) can be associated with Genero Application Server. With this system, you can start a 4GL application simply with a Java plug-in enabled browser and a URL.

See also : GJC Applet.

Installation

WARNING! GJC is installed into Genero Application Server (GAS). Therefore, Genero Application Server must be installed prior to installing the GJC module.

Windows Systems

To start the installation:

1. Close all applications.
2. Execute the installation program.
3. Follow the instructions provided by the installation program. When prompted, select the directory where GAS is installed.

X11 Systems

To start the installation:

1. Close all applications.
2. Execute the installation shell using the following command:
```
sh installation-script.sh -i
```
3. Follow the instructions provided by the installation program. When prompted, select the directory where GAS is installed.

After installation

See the *Genero Application Server Manual* for information on configuring the Genero Application Server and creating entries for GJC applications.

A demo application is available at the following URL:

Mode:	direct
URL:	`http://<server>:<port>/wa/r/gjc-demo`
Where:	<*server*> is the IP of the server <*port*> is the port used by the gas
Example:	`http://myServer:6394/wa/r/gjc-demo`

Mode:	cgi
URL:	`http://<server>[:<port>][<path>]cgi-bin/fglccgi/wa/r/gjc-demo`
Where:	<*server*> is the IP of the server <*port*> is the port of the web server (default 80) <*path*> is the path where the cgi is installed
Example:	`http://myServer/cgi-bin/fglccgi/wa/r/gjc-demo`

WARNING! You must use a URL using "`/wa/r`". In the Shortcut System, you must use a URL with "`/ja/r`". For more information on using URLs, see the Frequently Asked Questions section.

Template

During installation of the GJC archive, a basic template is installed. This template can be modified to fit your needs. For more information regarding template modification, see the *Genero Application Server Manual*.

The parameters specified in the basic template are specific to GJC applets. To configure the GJC template for IE browsers, the parameters are defined in <PARAM> tags. For Firefox/Mozilla browsers, all parameters defined in <PARAM> tags must also be defined in <COMMENT> <EMBED> tags. To be compatible with both IE and Mozilla-family browsers, we recommend that you define a parameter in <PARAM> and <EMBED> tags, as in the default template provided.

The following parameters can be modified freely:

- Parameters of the OBJECT tag, that loads the GJC:
  - `codebase="http://java.sun.com/update/1.5.0/jinstall-1_5_0_01-windows-i586.cab#Version=1,5,0,10"`:
    - Specifies the automatic JRE download path.
- `<PARAM>` tags:
  - `<PARAM name="codebase" value="/fjs/applet"/>`

        - Specifies the abstract path where the archive (usually **gjc.jar**; see below) can be downloaded.
    - `<PARAM name="resourcesPath" value="/fjs/applet"/>`
        - Specifies the abstract path where the resources (such as pictures) can be downloaded.
    - `<PARAM name="archive" value="gjc.jar"/>`
        - Specifies the name of the GJC archive. Usually **gjc.jar**.
    - `<PARAM name="admin" value="on"/>`
        - Set the administrator mode on or off.
    - `<PARAM name="debug" value="on"/>`
        - Set the debug mode on or off.
    - `<PARAM name="startApplicationOnLoad" value="on"/>`
        - Set the automatic launching of the application mode on or off.
        - If **on** is set, the GJC is loaded and the application specified in the URL is started.
        - If **off** is set, the GJC is loaded without any application.
- `<COMMENT>` tag:
    - All `<COMMENT>` parameters are also defined in the `<PARAM>` tags. Use the same value or your applet will not start.

GJC Functions

The following functions can be used on a GJC object to launch applications from a javascript:

- To start an application with parameters:

  ```
  startApp(applicationName, applicationQueryString)
  ```

  Notes:

  1. *applicationName* is the name specified in the configuration file of Genero Application Server.
  2. *applicationQueryString* represents all the application's parameters. If there are no parameters in your application, use an empty string.

- To start an application from a URL:

  ```
  startApp(url)
  ```

  Notes for direct mode:

  1. The URL is `http://<server>:<port>/ja/r/gjc-demo`
  2. <server> is the IP of the server.
  3. <port> is the port used by the GAS.

Example: `http://myServer:6394/ja/r/gjc-demo`

Notes for cgi mode:

1. The URL is `http://<server>[:<port>]/[<path>]cgi-bin/fglccgi/ja/r/gjc-demo`
2. <*server*> is the IP of the server.
3. <*port*> is the port of the web server (default 80)
4. <*path*> is the path where the cgi is installed.

Example: `http://myServer/cgi-bin/fglccgi/ja/r/gjc-demo`

Javascript functions

The default template contains several javascript functions which show how to use GJC in an HTML page:

`getGJC()`	Gets the GJC Object according to the navigator
`getDefaultURL()`	Returns the default URL
`startIt()`	Starts the default configured application in Genero Application Server

Wrappers used to launch an application:

`startApp(appName, appQueryString)`	Starts an application with an optional query string
`startUrl(p_url)`	Starts an application from a URL

In the <BODY> tag, a javascript statement has been added to warn the user before exiting the page (back, forward, reload, ...):

```
     onbeforeunload="return 'All currently running programs will also
be closed.'"
```

CSS Statement

The default template contains a CSS section for pretty menus.

HTML statement

The default template contains a menu to show the different ways to start an application when the applet is already loaded with one instance of GJC:

Template Example

```
<HTML>
    <HEAD>
        <TITLE>
            $(application.id) - Four J's Genero Java Client
        </TITLE>
        <META http-equiv="expires" content="1"/>
        <META http-equiv="pragma" content="no-cache"/>
<!-- CSS SECTION-->
<STYLE type="text/css">
#menu{
border: 1px solid black;
width: 170px;
background-color: #E6E6E6;
}
#menu a{
font: bold 13px Verdana;
padding: 2px;
padding-left: 4px;
display: block;
width: 100%;
color: black;
text-decoration: none;
border-bottom: 1px solid black;
}
#menu i{
color: blue;
}
#menu a:hover i{
color: aqua;
}
html>body #menu a{
width: auto;
}
#menu a:hover{
background-color: black;
color: white;
}
</STYLE>
<!-- SCRIPT SECTION-->
<SCRIPT type="text/javascript">
/**
 * Returns the default URL
**/
function getDefaultUrl() {
  return location.protocol + "//" + location.host + "$(connector.uri)"
+ "/ja/r/" +"$(application.id)"+"$(application.querystring)";
}

/**
 * Get the GJC Object according to the navigator
 * With Mozilla/Firefox embeds.length > 0, with IE applets.length > 0
**/
```

```
function getGJC() {
  if(document.embeds.length > 0) {
    return document.embeds.item('gjc');
  } else if(document.applets.length > 0) {
    return document.applets.item('gjc');
  } else {
    alert ("No Applet in this page");
  }
}

//Different way to start an application when the applet is already loaded

/**
 * Starts the default configured application
 **/
function startIt() {
  var gjc= getGJC();
  gjc.requestFocus();
  gjc.startApp("$(application.id)", "$(application.querystring)");
}

/**
 * Starts an Application with optional queryString
 * This function can be used with only one parameter (appName)
 **/
function startApp(appName, appQueryString) {
  var gjc= getGJC();
  gjc.requestFocus();
  getGJC().startApp(appName, appQueryString||"");
}

/**
 * Starts an application from a URL
 *
 * Mode: direct
 * URL: http://<server>:<port>/ja/r/gjc-demo
 * Where:   <server> is the IP of the server
 *          <port> is the port used by the gas
 * Example: http://myServer:6394/ja/r/gjc-demo
 *
 * Mode: cgi
 * URL: http://<server>[:<port>][<path>]cgi-bin/fglccgi/ja/r/gjc-demo
 * Where:   <server> is the IP of the server
 *          <port> is the port of the web server (default 80)
 *          <path> is the path where the cgi is installed
 *
```

```
 * Example: http://myServer/cgi-bin/fglccgi/ja/r/gjc-demo
 *
 **/
function startUrl(p_url){
  getGJC().startApp(p_url);
}

</SCRIPT>
    </HEAD>
    <BODY bgcolor="#FFFFFF" onbeforeunload="return 'All currently
running programs will also be closed.'">
        <H1>
            Application: $(application.id)
        </H1>
        <TABLE style="width: 100%; text-align: left;" border="0"
cellpadding="2" cellspacing="2">
            <TBODY>
                <TR>
                    <TD style="width: 200px; vertical-align: top;"><br>
                        <DIV id="menu">
                            <A HREF="javascript: startApp('gjc-
demo','');">Demo Application<br><i>using empty queryString</i></a>
                            <A HREF="javascript: startApp('gjc-
edit');">Edit Application<br><i>without queryString</i></a>
                            <A HREF="javascript: startApp('gjc-
toolbar','Arg=ToolBar1a');">ToolBar Application<br><i>using
queryString</i></a>
                            <A HREF="javascript:
startUrl(getDefaultUrl());">Demo Application<br><i>using direct
URL</i></a>
                            <A HREF="javascript: startIt();">Demo
Application<br><i>using startIt</i></a>
                        </DIV>
                    </TD>
                    <TD style="vertical-align: top;">
                    <CENTER>
                      <OBJECT
                        name="gjc"
                        classid="clsid:CAFEEFAC-0015-0000-0001-
ABCDEFFEDCBA"
codebase="http://java.sun.com/update/1.5.0/jinstall-1_5_0_01-windows-
i586.cab#Version=1,5,0,10"
                        width="450"
                        height="500">
                       <PARAM name="name" value="gjc"/>
                       <PARAM name="codebase" value="/fjs/applet"/>
                       <PARAM name="resourcesPath"
value="/fjs/applet"/>
                       <PARAM name="archive" value="gjc.jar"/>
                       <PARAM name="code"
value="com.fourjs.gjc.monitor.GJCApplet"/>
                       <PARAM name="type" value="application/x-java-
applet;jpi-version=1.5.0_01"/>
                       <PARAM name="mayscript" value="true"/>
                       <PARAM name="scriptable" value="false"/>
```

```
                        <PARAM name="applicationId"
value="$(application.id)"/>
                        <PARAM name="connectorURI"
value="$(connector.uri)"/>
                        <PARAM name="applicationQueryString"
value="$(application.querystring)"/>
                        <PARAM name="debug" value="on"/>
                        <PARAM name="admin" value="on"/>
                        <PARAM name="startApplicationOnLoad"
value="on"/>

                        <COMMENT>
                           <EMBED
                              name="gjc"
                              java_codebase="/fjs/applet"
                              resourcesPath="/fjs/applet"
                              archive="gjc.jar"
                              code="com.fourjs.gjc.monitor.GJCApplet"
                              type="application/x-java-applet;jpi-
version=1.5.0_01"
                              mayscript="true"
                              scriptable="false"
                              applicationId="$(application.id)"
                              connectorURI="$(connector.uri)"
applicationQueryString="$(application.querystring)"
                              debug="on"
                              admin="on"
                              startApplicationOnLoad="on"
pluginspage="http://java.sun.com/products/plugin/index.html#download"
                              width="450"
                              height="500">
                           </EMBED>
                        </COMMENT>
                     </OBJECT>
                  </CENTER>
               </TD>
            </TR>
         </TBODY>
      </TABLE>
   </BODY>
</HTML>
```

Java Web Start Administration

This section is directed at system administrators and server administrators.

- Overview
- Application Server Deployment
- Web Server Deployment
- Upgrade
- Multi version
- Arguments
- Dynamic JNLP

Overview

Java Web Start (JWS) technology is a network deployment and upgrade system that can be associated with a Web browser.
JWS relies on Java Network Launching Protocol (JNLP) and API Specification, v1.0. For more information about JWS, visit http://java.sun.com/products/javawebstart.

Genero Java Client is compatible with Java Web Start technology provided with SUN Java Runtime Environment and can be installed and launched by a Web browser using a hyperlink.
Once installed, the GJC can be launched by using a shortcut, even if it is disconnected from the server. If it is connected to the server, JWS checks if the GJC is up to date and upgrades if needed.

Unlike Java applets, Genero Java Client is downloaded into the client and started as a stand-alone application. As a result, all stand-alone features are enabled (such as command line arguments and monitor options).

Application Server deployment

The default deployment server for GJC over JNLP is the Genero Application Server (GAS), however most Web servers may be used.
When installed in the GAS (with the applet install package), GJC is provided with this default **$GASDIR/web/fjs/jws/gjc.jnlp** file for Java Web Start deployment:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+"
      codebase="http://host:6394/fjs/jws/"
```

```
      href="gjc.jnlp">

  <information>
    <title>Genero Java Client</title>
    <vendor>Fourj's Development Tools</vendor>
    <homepage href="http://www.4js.com"/>
    <description>Genero Java Client</description>
    <description kind="short">Genero Java Client (GJC) is a Graphical
Front End for Genero Runtime System.</description>
    <icon href="ico_4js.png"/>
    <offline-allowed/>
    <shortcut online="false">
      <desktop/>
      <menu submenu="FourJ's GJC version"/>
    </shortcut>
  </information>
  <security>
      <all-permissions/>
  </security>
  <resources>
    <j2se version="1.4.0+"
href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="/fjs/applet/gjc.jar"/>
  </resources>
  <application-desc main-class="com.fourjs.gjc.monitor.Monitor"/>
</jnlp>
```

The **gjc.jnlp** file describes how GJC is downloaded, installed and launched on the client system. To modify the JNLP shortcut behavior, modify this file. For more information, refer to http://java.sun.com/products/javawebstart.

Web Server deployment

Many Web servers do not provide out-of-the-box support the JNLP mime type. For such Web servers, you must add the JNLP mime type to the default configuration.

Apache 2 Web server example

For an Apache 2 Web server, you add the following line to the **mime.types** file:

```
      application/x-java-jnlp-file    jnlp
```

Choose a codebase for your JNLP install. For this example, assume the codebase is "/fjs".

Put the **gjc.jar** file in **<codebase>/gjc** (**/fjs/gjc/gjc.jar**).

Put the icon in <codebase> (**/fjs/ico_4js.png**).

Copy the default **gjc.jnlp** file into **<codebase>/jnlp** (**/fjs/jnlp/gjc-*version*.jnlp**).

Edit the jnlp file:

```
...
     codebase="http://host[:port]/<codebase>"
     href="jnlp/gjc-version.jnlp"
...
     <jar href="gjc/gjc.jar"/>
```

At this point, Java Web Start deployment for GJC should work. Insert the JNLP hyperlink into an HTML document using an anchor tag:

```
<a href="http://host[:port]/fjs/jnlp/gjc-version.jnlp">mylink</a>
```

Upgrade GJC versions

If you need to upgrade all clients to a new GJC version, modify the jnlp file to point to another java executable file (**gjc_*version*.jar**), or just install the new GJC applet into your GAS.
Any client connecting to your GAS using JWS will automatically upgrade to and start with the new version. Clients disconnected from the network, however, will not be upgraded.

Note: By default, the upgrade process relies on the "Last-Modified" http response header. You can override this default behavior by specifying a version to be used. If a version is set, the "Last-Modified" http response header is ignored. For example:

```
<jar href="/fjs/gjc.jar" version="1.32.1a"/>
```

Maintain multiple GJC versions

If you need to maintain several GJC versions at the same time, you should create one jnlp file for each group of users. The version numbers in the examples are for illustrative purposes only:

Example 1: by Genero version

create one **gjc-<*version*>.jnlp** for each version used in your organization and add the java executable file **gjc-<*version*>.jar** into your server.
gjc-1.32.jnlp (points to 1.32.1a)

gjc-1.40.jnlp (points to 1.40.1a)

When updating a version of GJC 1.32, modify the corresponding JWS file :
gjc-1.32.jnlp (now points to 1.32.1b)
gjc-1.40.jnlp (still points to 1.40.1a)

Example 2: by user profile

gjc-users.jnlp (points to 1.32.1a)
gjc-old.jnlp (points to 1.30.1f)
gjc-latestbuild.jnlp (points to 1.32.1b-211)

When updating a version of GJC, modify the corresponding JWS file :
gjc-users.jnlp (now points to 1.32.1b)
gjc-old.jnlp (points to 1.30.1f)
gjc-latestbuild.jnlp (now points to 1.32.2a-212)

Note for GAS users: **GAS is set to provide one GJC at a time. If you want to maintain more than one version, please copy the** fjs/applet/gjc.jar **file into** fjs/applet/gjc-<*version*>.jar **each time you install a new GJC applet. and use a copy of the provided** gjc.jnlp **modified to use the** gjc-<*version*>.jar**.**

Provide arguments

Command line arguments are available within a JWS deployment. To add a command line argument, you simply add an `<argument>` child to the `<application-desc>`:

```
<application-desc main-class="com.fourjs.gjc.monitor.Monitor">
   <argument>-aD</argument>
   <argument>--port=6500</argument>
<application-desc/>
```

Generate dynamic jnlp files

You can generate the jnlp file by using your preferred script language in a web server to give command line arguments to the GJC. You have to set the http content-type header field, as well as the codebase and href location, so that href is the real URL. In addition to those three requirements, you can further modify the jnlp file as needed.

Example

Use php to add a --url argument (http connector launch), retrieving the argument in a JNLP url such as:

```
  http://<codebase>/
gjcversion_number.php?url=http://gashost:6394/ja/r/gjc-demo
```

```
<?php
  header('Content-Type: application/x-java-jnlp-file');
  header('Pragma: no-cache');

  echo '<?xml version="1.0" encoding="utf-8"?>';

  echo '<jnlp spec="1.0+"';
  echo '   codebase="http://host/<codebase>/" ';
  echo '   href="http://neo/~og/jnlp/gjcversion.php?url=';
  echo $_GET['url'];
  echo '">'
?>
  <information>
    <title>Genero Java Client version</title>
    <vendor>Fourj's Development Tools</vendor>
    <homepage href="http://www.4js.com"/>
    <description>Genero Java Client</description>
    <description kind="short">Genero Java Client (GJC) is a Graphical
Front End for Genero Runtime System.</description>
    <icon href="ico_4js.png"/>
    <offline-allowed/>
    <shortcut online="false">
      <desktop/>
      <menu submenu="FourJ's GJC version"/>
    </shortcut>
  </information>
  <security>
      <all-permissions/>
  </security>
  <resources>
    <j2se version="1.5.0+"/>
    <jar href="gjc/gjcversion.jar"/>
  </resources>
  <application-desc main-class="com.fourjs.gjc.monitor.Monitor">
    <argument>-u <?php echo $_GET[url]; ?></argument>
  </application-desc>
</jnlp>
```

Java support in Netscape browsers

GJC client needs a Java Runtime Environment installed in the system to work with JWS. If you want to check if java is enabled on the client system, refer to http://java.sun.com/products/javawebstart

JNLP support

Java Web Start primarily supports Internet Explorer 4 or higher and Netscape 4.X. However jnlp files can be launched from any browser if the mime-type association is done correctly. Please note that Java Web Start uses the browser's settings and may launch a browser to show a URL these may/may not work with unsupported browsers.

Java Web Start works with Netscape 6. However, you will need to register Java™ Web Start manually with NS6. This is done in the Navigator/Applications section of the Navigator/Helper Application section.

Browser shows jnlp file as plain text

This is most likely happening because your web server is not aware of the proper MIME type for jnlp files. Java Web Start requires only one change to your web server, that is creating an association between the file extension, typically JNLP, and the mime type, `application/x-java-jnlp-file`. The steps for doing this vary depending upon the Web server you are using.

Java Web Start for End Users

This section is directed at end users.

- JWS Overview for End Users
- Java console
- Java control panel

JWS Overview for End Users

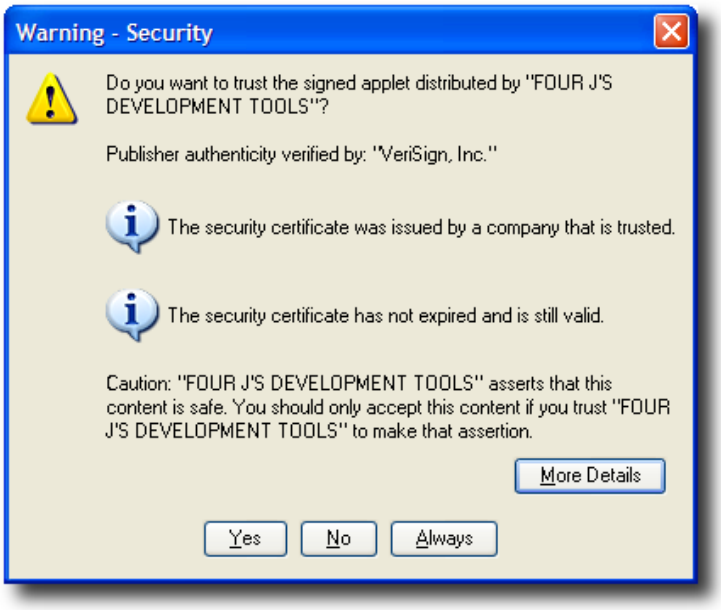
When you click on the GJC jnlp link, a dialog informs you that Java is downloading/upgrading the software.

Depending on the configuration options set by the administrator, when you access the GJC jnlp link for the first time, a dialog allows you to create a shortcut on your desktop and start menu. In addition, you can choose to open the java control panel.

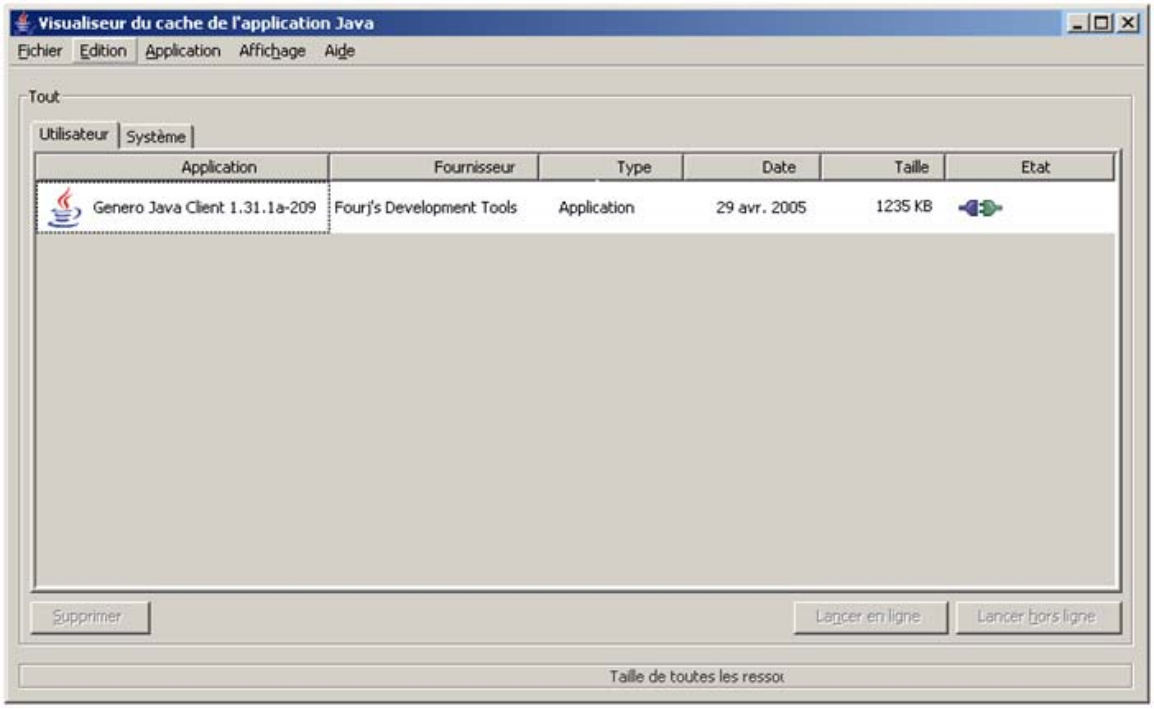
Next, the GJC is launched, with or without a 4GL application. The connection using GJC jnlp may be "http" or "direct" depending on the shortcut.

A window may ask you if you accept the Four J's security certificate (if it is not already present in your certificate store). We recommend that you add the certificate to your certificate store by clicking the **Always** button, so that this window will not appear each time you launch a Four J's trusted application.

Note: The Four J's security certificate is verified by Verisign (http://www.verisign.com).



The GJC is stored in the Java cache. You can access the Java cache using **javaws** tools (**<JRE_PATH>/bin/javaws**) to remove the GJC installed with JWS.



Java console

The Java console is not useful with GJC. You can close the console window.

Java control panel

The Java setup panel manages the Java network, security, update, and many other parameters. We recommend that you leave the default parameters or follow the recommendations of your system administrator.

For more information on the Java control panel, refer to http://java.sun.com/j2se/1.5.0/docs/guide/deployment/deployment-guide/jcp.html.

Security

By default, GJC accepts all the connections that arrive on the listening port, without any verification. The command Line option "`-A`" adds some security checks.

- Security Level 1
- Security Level 2 and 3

Security Level 1

Command Line : `java -jar gjc.jar -A 1`

With security level 1, when the Runtime System starts a connection, the GJC user is warned by a pop-up.

- If you select "`Yes`", GJC will accept this (and only this) connection. Any other connection attempt will raise the pop-up again.
  If you select "`Always`", GJC will accept this connection and any other connection from the same host. This will be saved when GJC is closed and will be reapplied when it is restarted.
  If you select "`No`", GJC will reject this connection; the application will not run on the front end.

"Always" authorized hosts are saved in the **$GJCDIR/etc/hosts.xml** file. You can modify this if needed, or remove this file if you want to reset the authorized list.

Security Level 2 and 3

Command Line : `java -jar gjc.jar -A 2` or `java -jar gjc.jar -A 3`

Warning! This only works when using a direct shortcut to start an application.

Security levels 2 and 3 use the following mechanism:

1. When started, GJC generates a random key.
2. When you start a shortcut, @FGL (and @FGLxxx) will set this key in the `_FGLFEID` environment variable.
3. When `fglrun` starts, it reads the environment variable and sends it back to the GJC.
4. GJC compares the internal key and the key sent back by the Runtime System. It only accepts a connection with the same key. For all others:

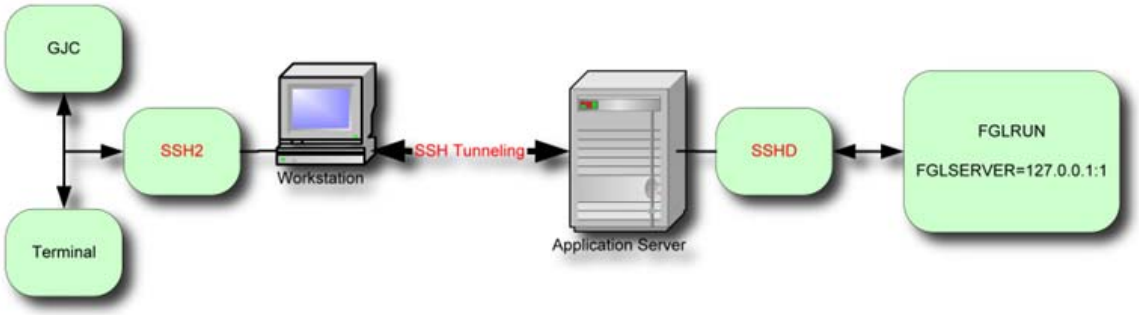
- Security level 2 display a pop-up and asks if a wrong key connection can connect. You choose whether the connection occurs or is rejected.
- Security level 3 will reject the wrong key connection.

Warning! If the Runtime System you are using does not handle this feature, you won't be able to run an application in this security level.

GJC and SSH

Summary:

Overview



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 more feature it offers, which will be explored here, is port forwarding.

Port forwarding allows an application on one computer to connect to a local port and have the data tunneled through the ssh session to the other computer. The advantage to this is that it doesn't require you to open any more ports on your firewall router other than one you already have open to ssh. If you have firewalls, this is an advantage because Genero needs to establish a connection going from the client to the server to start the user application and another connection 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.

Any environment that uses firewalls or connections over the Internet should use ssh or ssh2 for the connections. You should never send unencrypted data such as account numbers, social security numbers, and passwords through the Internet without encryption. Some companies might even consider this for internal use when you consider accounting and payroll information. At any point along the way, someone could be

monitoring the data for malicious intent or even network diagnostics. Whatever the reason, encryption is simple now and offers peace of mind.

SSH is comprised of two main components, the server component "sshd" and the client component "ssh". With Genero we have provided our own built-in client component.

Prerequisites

Things you should know about your system:

In order to determine what is needed to proceed with this document, you 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, as this topic will not be covered in this document.

How do I make sure data is encrypted?

If you are using the shortcut buttons in the GJC, there are two connections established between the client and server. The first is established from the client to the server to log in and start the application. The second is from the server's application to the client to display the graphical data.

Use the chart below to determine which settings you will need.

Type of connection	Command encrypted	GUI encrypted
SSH2		
SSH2 port forwarding		

What connection method should I use?

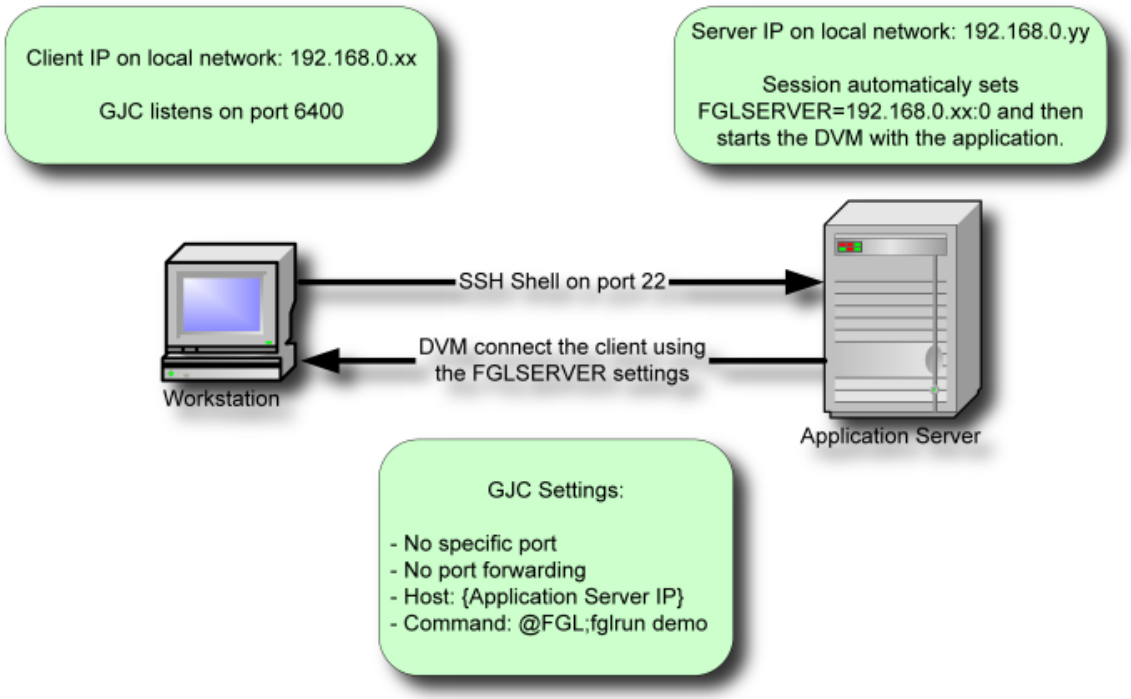
Knowledge of your configuration will be necessary to make Genero work properly. Refer to the Prerequisites section to find out what information will help.Use the matrix below to determine which connection methods will support what you are trying to do. Currently the SSH2 with Port Forwarding provides the most flexible connectivity.

	SSH2	SSH2 + Port Forwarding
Firewall or NAT on Server Side	1	1
Firewall or NAT on Client Side	2	
Firewall or NAT on Both Sides	1,2	1
Private Network		
VPN (Same as Private Network)		
Encryption of all Data		
Password/login Encrypted		
Keep Alive		, 3

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 GJC.
3 - May require changes to firewall connection timers if firewalls are involved.
- Indicates full functionality with no changes.

GJC and SSH: Simple Setup

The simple setup assumes that you are on a corporate LAN with no firewalls. SSH2 will work fine, without any special set up 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.



What if you want to connect to another port other than 6400 for the GUI? You can specify on the command line for GJC the option "-p <port> " and the GJC will listen on that port for the GUI connections. The FGLSERVER will have its information adjusted accordingly. For example, execute "java -jar gjc.jar -p 7400". When you look at the value of FGLSERVER it will contain "<client IP> :1000". If it was the default of port 6400 it would be "<client IP> :0" because the default is 6400 and the number after the colon is added to this number.

If you do port forwarding while using "-p 7400" on the GJC 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 GJC's running at the same time, refer to section 6 for details on how to do this.

Port Forwarding and Firewalls

Port Forwarding

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

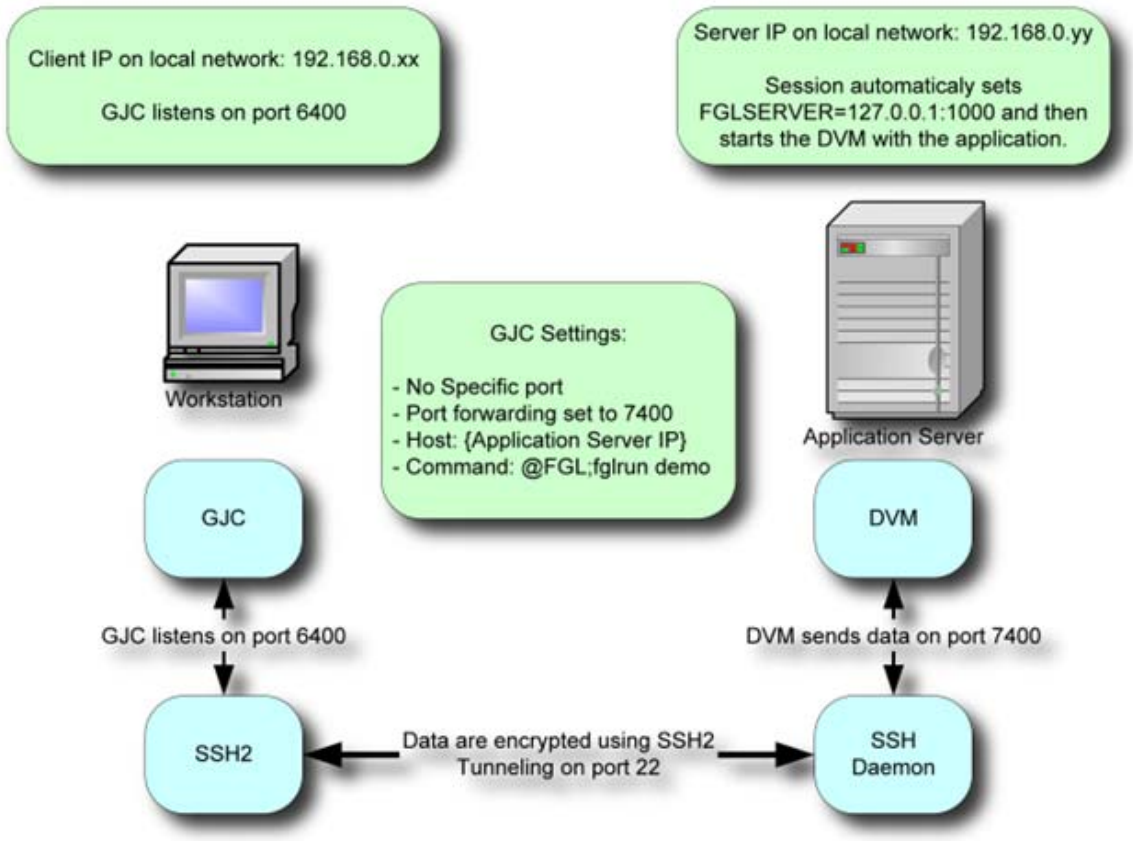


Figure 1

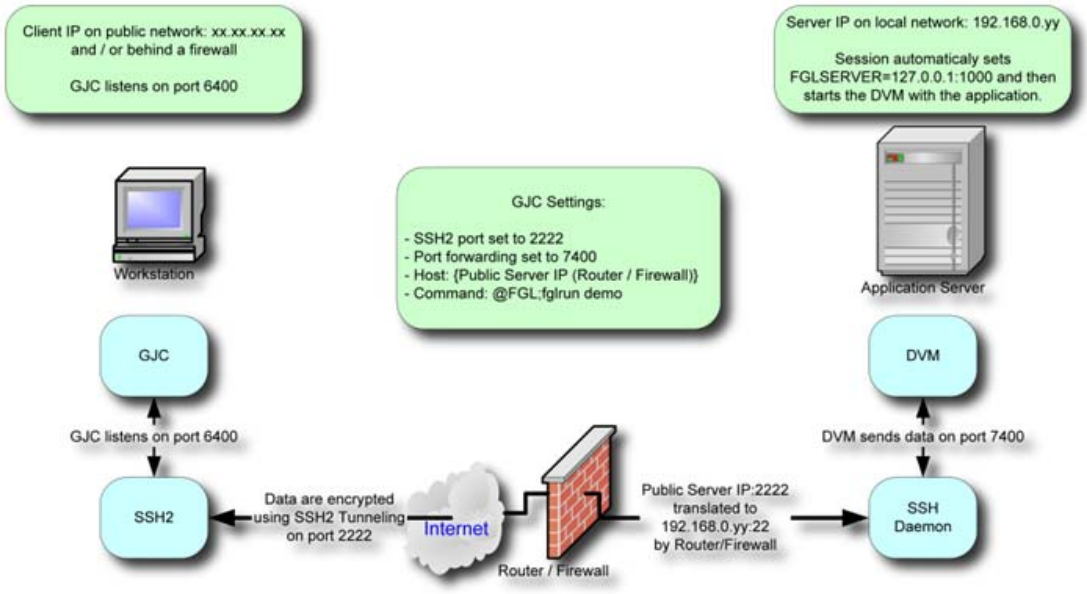


Figure 2

Figure 1 shows a simple configuration that does not involve a firewall. SSHD, the portion running on the server will accept a connection from the GJC (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 tunnelled through the existing connection to the client, which will display the application. Note that both sides still use ports to accomplish this.

You will need to have ssh installed and set up on the server. If you are expecting to access your Genero application from somewhere on the Internet then you will most likely have a firewall router and have to open a port on your router to allow connections to the SSHD. Refer to figure 2 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 is what you will need to set in the GJC using the "Specific Port" field.

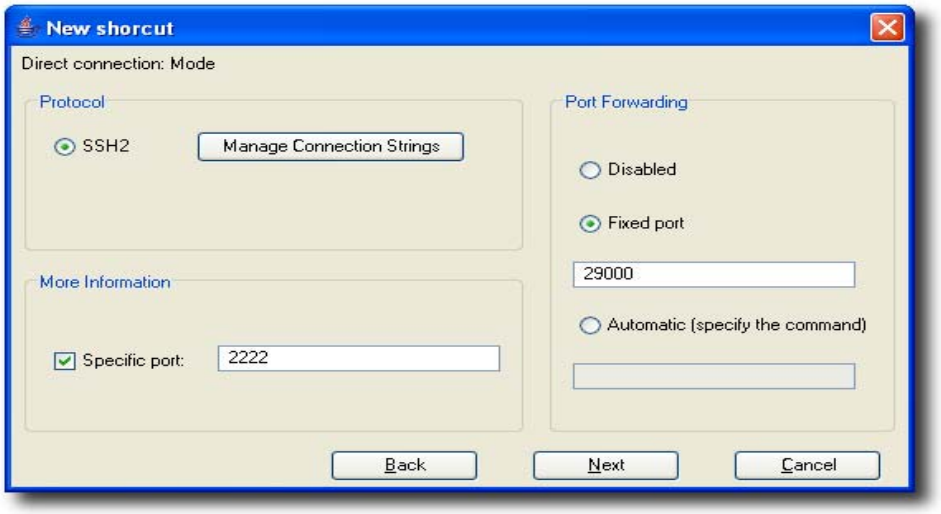


Figure 3

In figure 2 we have set our firewall router to forward port 2222 on 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 will have to use different port numbers and map each server with a different port number. Most routers will allow the destination port to be different than the origination port. An example is that 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 in the Firewall Server Side section.

In figure 3 we have also set Port Forwarding to 29000. This will cause the SSHD running on the server to listen to port 29,000 for connections from the application. The FGLSERVER environmental variable will be set to 'localhost:22600'. It is localhost because it will be tunnelled and SSHD is running on the same machine. The 22600 is an offset for the port. To clarify, Genero GJC 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 an application that will return a free port number. As this application is really depending on the system the Runtime System is installed, we can't provide a version for each system. This program must be used in combination with the GJC connexion strings system.

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 then the IP address should be set to that of the firewall router. Refer to the line labelled "Command Line". Here is where you need to put @FGL so that Genero can set the FGLSERVER variable for you when it logs into the server. It 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 sucession. In Unix you put a semi-colon (;) and Windows you use two ampersands (&&) followed by whatever commands are necessary to start your application.

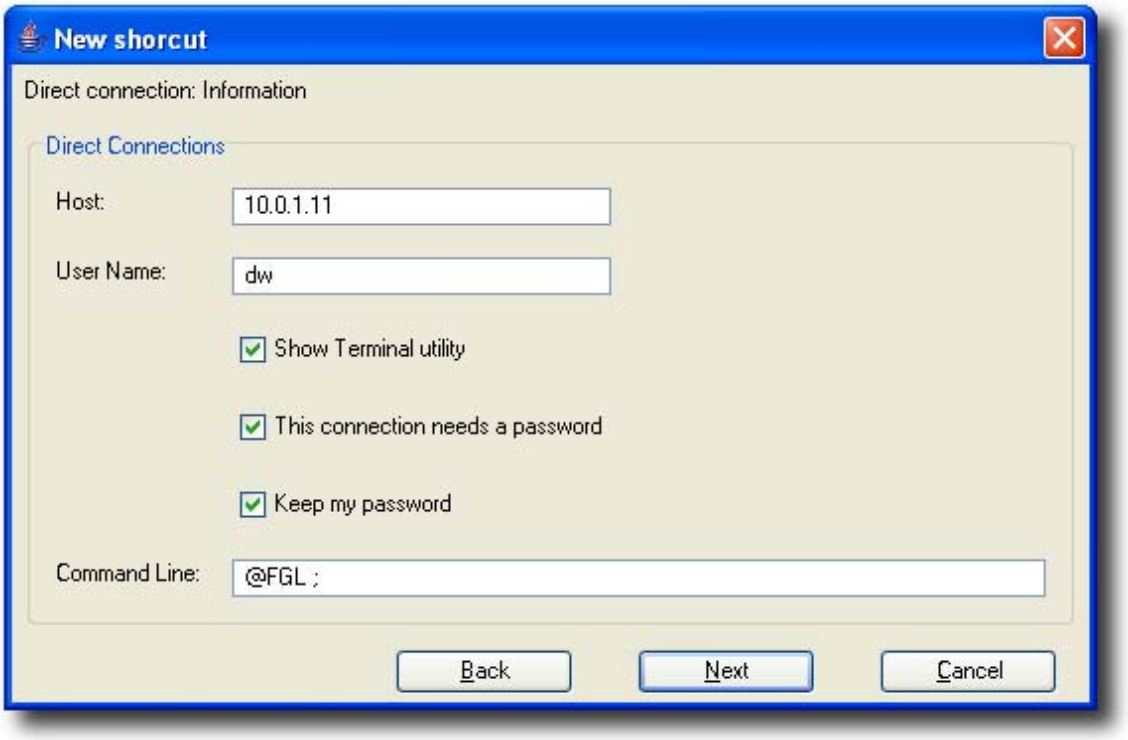


Figure 4

Firewalls : Client Side

This section will detail how to configure the client side. It will not go into what is required on the server side or how to set up port forwarding as that is covered in another section. Having a client side firewall means that 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 SSH2. This is by far the easiest and most secure method to connect without the help of a VPN:

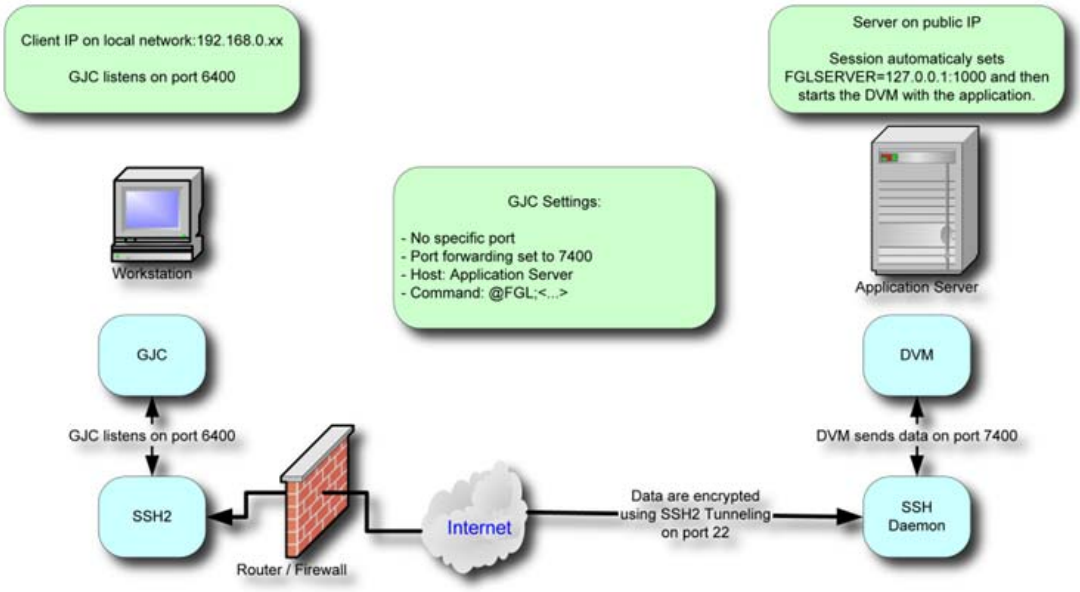


Figure 5.1

The second method requires adding rules to the router to allow connections. Since port forwarding is covered in a separate section, the set up of the router will be covered here:

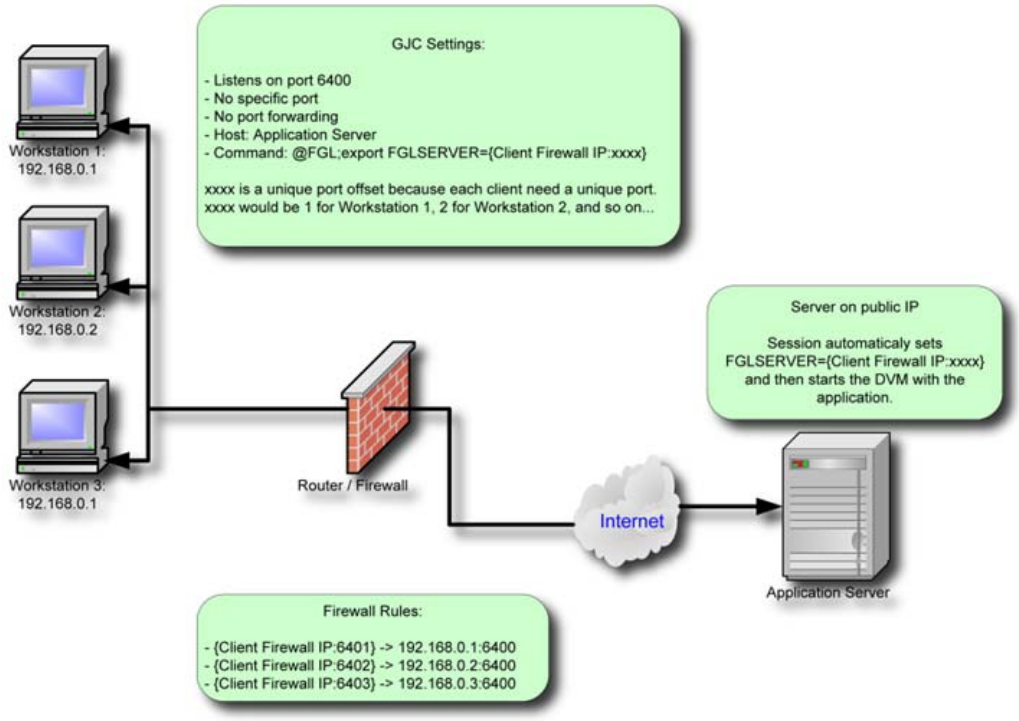


Figure 5.2

The router will need rules added to take a connection coming in on a specific port and directing 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: The examples are shown where the internal addresses are not public IP addresses. If you have public IP addresses on each client then you can open port 6400 for each of the clients.

Example rule:

```
Incoming 6400 -> 192.168.1.10:6400
```

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

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

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

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

This last example will require you to start the GJC with the -p option to cause them to listen on a different port than the default.

```
>java -jar gjc.jar -p 6401
>java -jar gjc.jar -p 6402
```

If you are setting up multiple clients in this manner, you may want to avoid starting the first client on 6400 because any mis-configured new clients will pop up on that users console unexpectedly.

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

Let's say the client firewall router's IP address to the Internet is 213.39.41.73 and port 10,000 is mapped to the client 192.168.0.53, port 6400. The entry in the router would be:

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

The command line in the GJC would look like this:

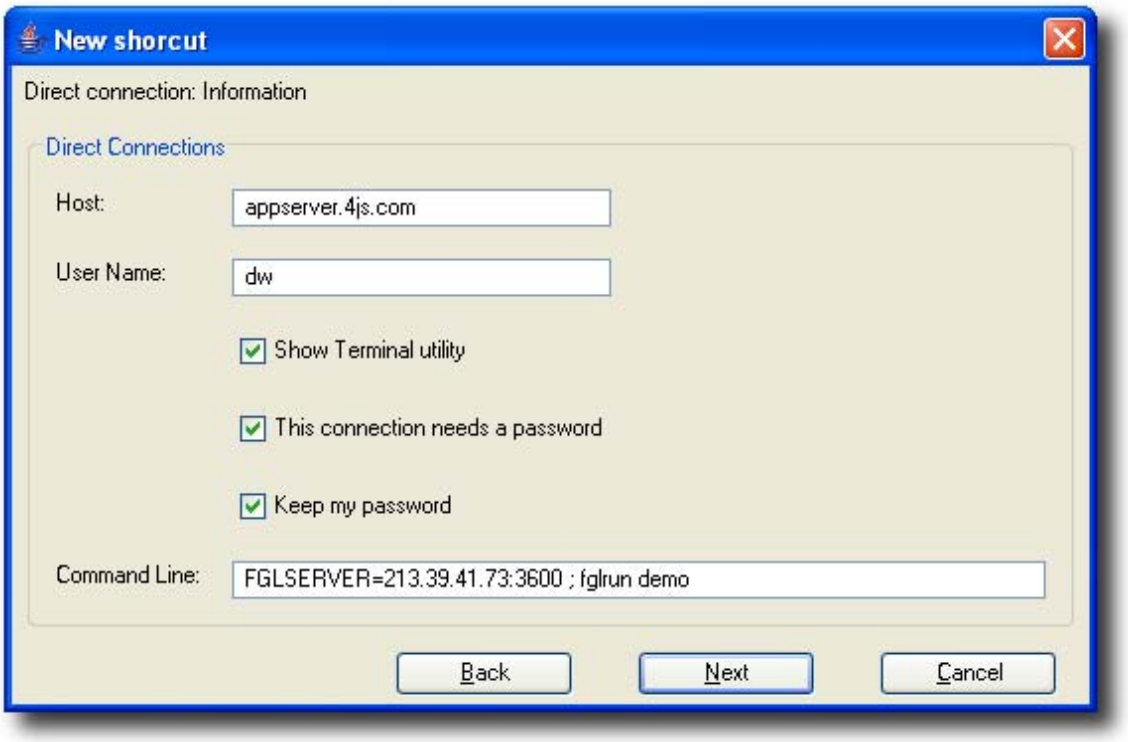


Figure 5.3

The FGLSERVER variable is normally set using @FGL but that would set FGLSERVER to be the IP of the local client machine and whatever port the GJC was started with using -p. If the IP addresses used behind the firewall are public, then this would be ok. But if the addresses are not public then 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 needs to be set to 10000 subtract 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.

Firewalls : Server Side

Having a server side firewall is the typical configuration on many systems. There is only one method for doing this regardless of whether you use telnet or SSH. That is to map a port to be forwarded to the server in the firewall router. It is not advised to use rlogin from the Internet for security reasons and 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 on 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.

Method using a telnet type connection:

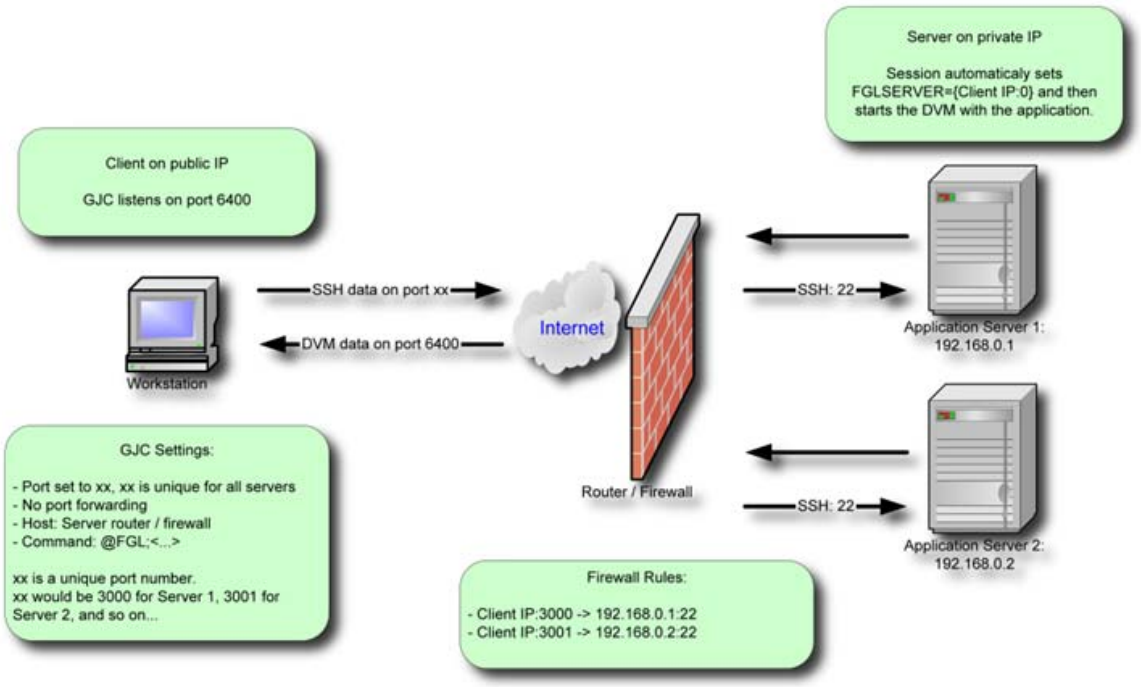


Figure 6.1

Notice that the returning GUI path doesn't require any special handling unless there is a client side firewall. For details on this refer to the Client Firewall section.

Method using SSH with port forwarding:

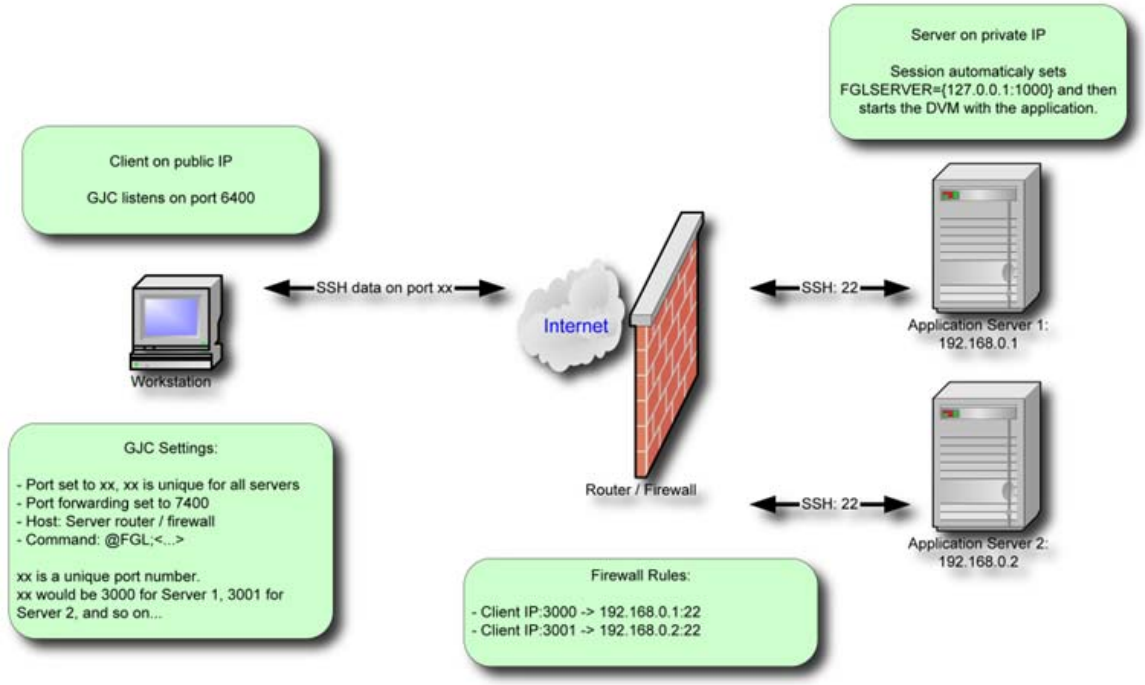


Figure 6.2

The client GJC would connect to the server firewall router on port 3000 to access server 1 and port 3001 for server 2. These ports we arbitrarily chose and pretty much 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 tunnelled 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 will need to be used if there are 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 please read the section on client side firewall configuration.

Let's look at an example. We have two servers that will be accessed via clients somewhere on the Internet. They will use SSH2 with port forwarding to simply 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 GJC client will need to be configured as well:

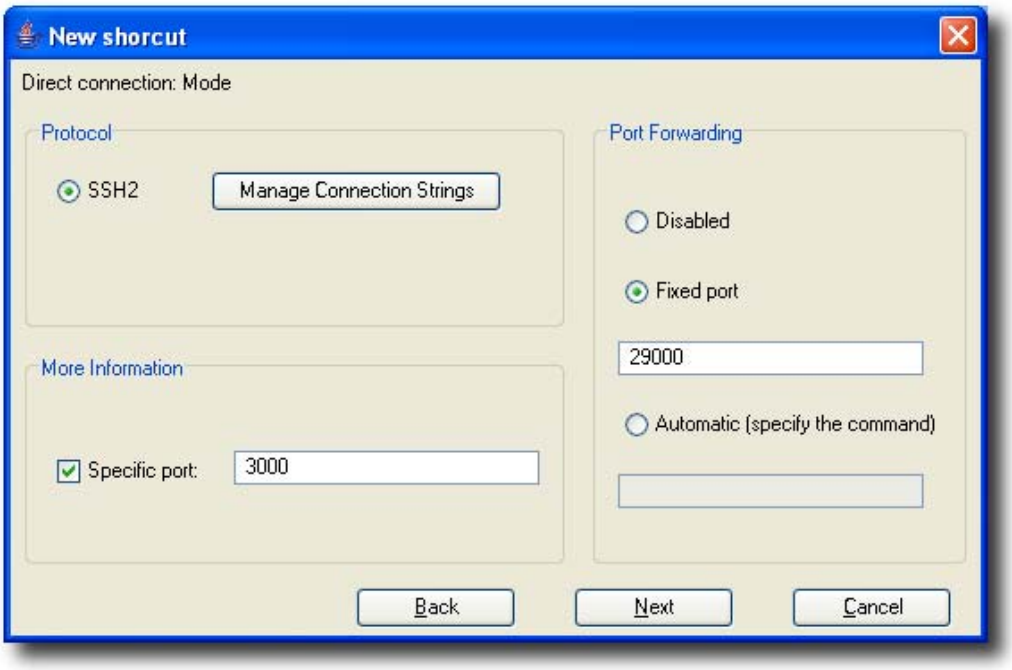


Figure 6.3 Showing configuration for access to Server 1

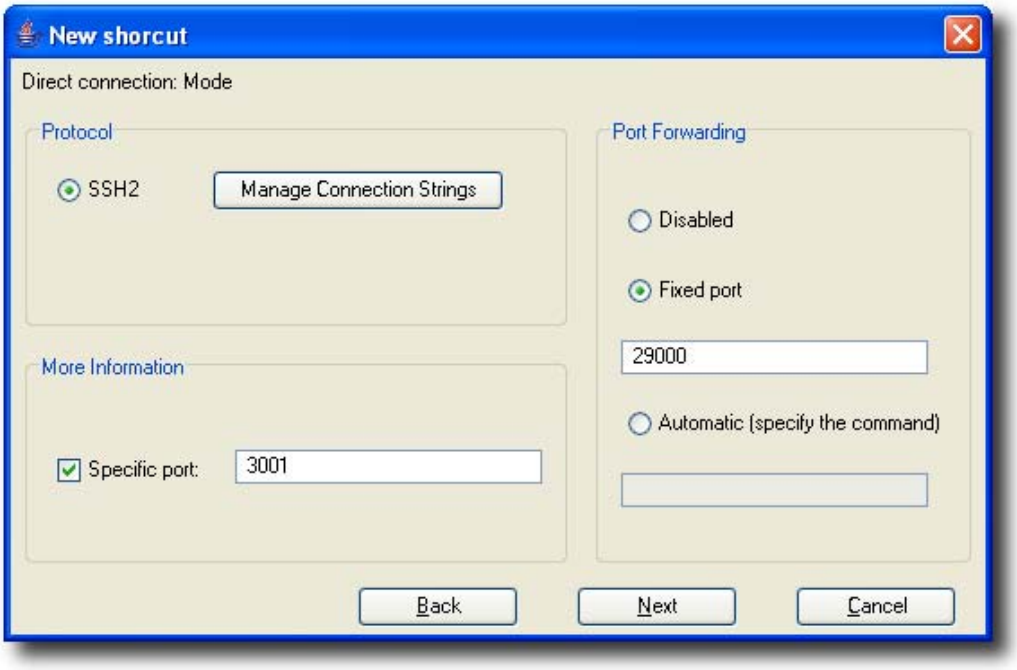


Figure 6.4 Showing configuration for access to Server 2

Figures 6.3 and 6.4 show how to access each server by specifying the appropriate port for each, one with 3000, the other 3001. This will allow the firewall router on the server side to direct each to the appropriate server. The IP address used would be the IP of the router.

Keep in mind that if you have two users accessing the same server, you will need to manually select a different port forward number to keep them unique. Refer to the section Possible configuration problems.

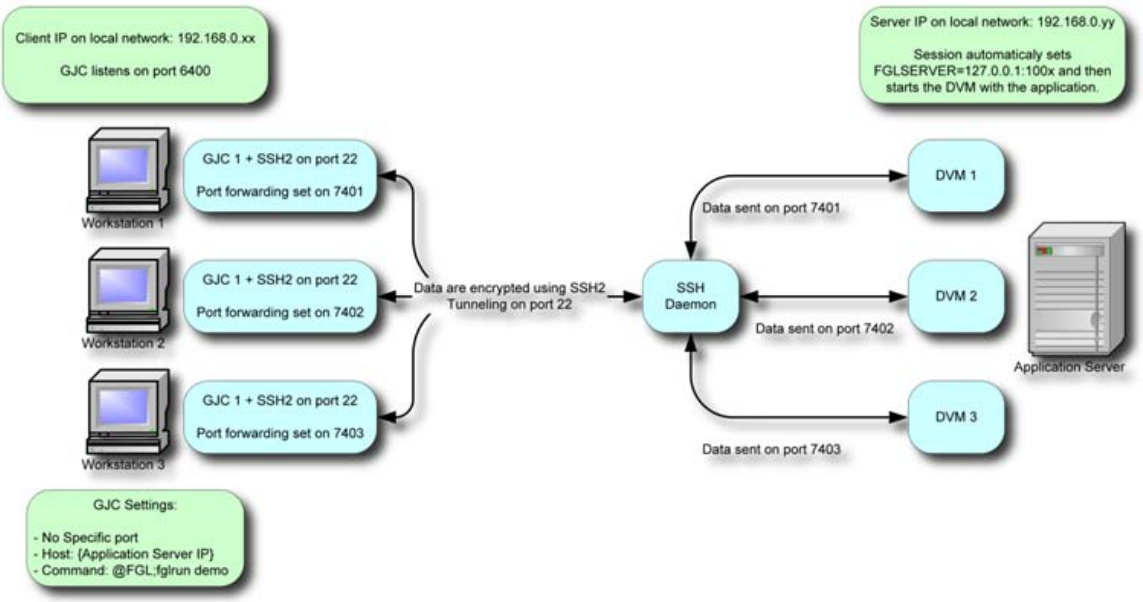
Possible Configuration Problems

Summary:

- Duplicate port forward number
- Wireless
- Need to change the port that GJC listens on
- Sessions Expiring

Duplicate port forward number

A situation that could become a problem if you are using Port Forwarding is if you have more than one client using the same Port Forward port number. What could happen is the application display could go to the wrong client. This is an expected result so precautions need to be taken to prevent this. Make sure that each client using port forwarding to a particular server uses a different port forward port number.



What is happening is the SSHD sets up a tunnel and listens on the port that is specified in the port forward field. This allows the Genero application (or DVM) to connect to that port and have the GUI data sent to the client. Only one listener can be listening on any port at a time. Each client needs to use a unique port number to avoid any problems.

NOTE: An automatic configuration solution is being worked on but has not been released at the time of this writing.

Wireless

A pitfall to be aware of is with wireless systems. The latest technology is to use 802.11(a,b or g). These are great at avoiding the wire mess but imply a new risk. Under Windows while using a wireless card plugged in or built in, if the signal is lost even for a second, the interface goes off line. When this happens it is treated similar to unplugging your network cable. The Windows drivers handle this by reporting to the network stack that the interface is now off line and everything associated with that interface is removed. If an application has an open channel, it is signalled that it has closed. The end result is you lose all your connections, wait for your signal to return and start logging in all over.

A possible work around for this problem is to use an external wireless device that doesn't take the connection down when the signal is lost. This would work 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 at before.

Need to change the port that GJC listens on

Why would you want to change the port you listen on? You may have a need to run several versions of the GJC on the same machine. Since each one needs to have its own listening port, Genero allows you to specify the port. If you ran more than one and didn't specify the port, Genero would open the next available port. For example the first instance would open 6400, the next instance would open 6401.

```
>java -jar gjc.jar              <- The port assigned would be 6400
>java -jar gjc.jar  -n          <- The port assigned would be 6401
>java -jar gjc.jar -n -p 7400    <- The port assigned would be 7400
>java -jar gjc.jar -n -p 7400    <- The port assigned would be 7401
```

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 then add rules to the router to send 6400 to the first client, 6410 to the second client and so on. Then each client would be started with the corresponding -p <port> and the router would make sure each client gets the connections intended for it.

Sessions Expiring

If you are having sessions expire or applications that disappear, you will need to check for routers that expire sessions. Most likely there is a firewall router in the path. If you

are using a firewall router you should check for session expiration timers for the ports used to get through the firewall. The expiration duration (aka KeepAlive) should be set to be 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 to conform to this. Use a value of 2 hours and 10 minutes.

GJC and Windows XP Service Pack 2

Summary:

- Firewall configuration

Firewall configuration

Microsoft has added several security systems in Windows XP Service Pack 2 (SP2). The firewall included in Windows XP has been improved and is now enabled by default. From the network point of view, GJC is a server: it listens on a defined port (6400 by default) for Runtime System connection.

When GJC starts, the firewall detects it listens on port 6400 and warns the user:

Click the **Unblock** button to allow GJC to work correctly.

Warning! Clicking on the **Keep Blocking** or **Ask Me Later** buttons will keep GJC from working correctly. Connections from the Runtime Systems will be blocked by the firewall.

If the **Keep Blocking** button was clicked by mistake, this parameter can be changed in the Firewall settings (Control Panel) in the "Exceptions" tab. javaw and/or java (depends if you launch the GJC with command line or not) must be present and checked.