Sun™ Shared Visualization 1.1 Software Client Administration Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 820-3257-13
November 2008, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Please
Recycle

Contents

Figures

Tables

Preface

This client administration guide provides detailed information and procedures for starting and using the Sun™ Shared Visualization 1.1 software (including 1.1 Update1 and 1.1.1 releases). This document is intended for users who are proficient in shell tool activities, such as system administrators, and people who have advanced experience with the Solaris™ Operating System, and other computing platforms.

Before You Read This Document

To fully use the information in this document, you must be familiar with the following software packages:

- Sun Grid Engine (if your site is using it)
- X11

How This Document Is Organized

Chapter 1 introduces the Sun Shared Visualization 1.1 software and how the software interacts with other software packages.

Chapter 2 discusses installation information for a Sun Shared Visualization 1.1 client.

Chapter 3 describes procedures for manually starting the Sun Shared Visualization 1.1 software.

Chapter 4 describes how to use Sun Grid Engine to start the Sun Shared Visualization 1.1 software.

Chapter 5 explains how to use the Advance Reservation feature.

Appendix A provides reference information about VirtualGL options and environment variables.

Appendix B provides basic reference information about TurboVNC.

Appendix C provides basic information about the Sun Grid Engine commands and options. It also provides a sample Sun Grid Engine job script that can be edited for your specific use.

Note – In this document these x86 related terms mean the following:
"x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
"x64" points out specific 64-bit information about AMD64 or EM64T systems.
"32-bit x86" points out specific 32-bit information about x86 based systems.

Using UNIX Commands

This document might not contain information about basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices. Refer to the following for this information:

- Software documentation that you received with your system
- Solaris Operating System documentation, which is at:

  `http://docs.sun.com`

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type `rm` *filename*.

Note – Characters display differently depending on browser settings. If characters do not display correctly, change the character encoding in your browser to Unicode UTF-8.

Related Documentation

Application	Title	Part Number	Format	Location
Getting Started	*Sun Shared Visualization 1.1 Software Getting Started Guide*	820-0237	Printed PDF	Shipping kit Online
Server Administration	*Sun Shared Visualization 1.1 Software Server Administration Guide*	820-3256	PDF	Online
Release Notes	*Sun Shared Visualization 1.1.1 Software Release Notes*	820-3258	PDF	Online
Sun Grid Engine	*N1 Grid Engine 6 Administration Guide*	817-5677	PDF	Online
Sun Grid Engine	*N1 Grid Engine 6 Release Notes*	817-5678	PDF	Online
Sun Grid Engine	*N1 Grid Engine 6 User's Guide*	817-6117	PDF	Online

Application	Title	Part Number	Format	Location
Sun Grid Engine	*N1 Grid Engine 6 Installation Guide*	817-6118	PDF	Online
Sun Grid Engine	*Sun Grid Engine Information Center* (http://wikis.sun.com/display/GridEngine/Grid+Engine)		HTML	Online
VirtualGL	*VirtualGL 2.1 User's Guide* www.virtualgl.org/Documentation/Documentation		HTML	Online

The *VirtualGL User's Guide* is also present on any system with Sun Shared Visualization 1.1 software (or VirtualGL) installed:

- On Solaris systems and all systems with Sun Shared Visualization 1.1.1 or VirtualGL 2.1.1, in the `/opt/VirtualGL/doc/index.html` directory
- On Linux systems with Sun Shared Visualization 1.1 software or VirtualGL 2.1 installed, in the `/usr/share/doc/VirtualGL-2.1/index.html` directory

Documentation, Support, and Training

Sun Function	URL
Documentation	http://www.sun.com/documentation/
Support	http://www.sun.com/support/
Training	http://www.sun.com/training/

Third-Party Web Sites

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun is not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

http://www.sun.com/hwdocs/feedback

Please include the title and part number of your document with your feedback:

Sun Shared Visualization 1.1 Software Client Administration Guide, part number 820-3257-13

CHAPTER 1

Sun Shared Visualization 1.1 Introduction

This chapter introduces the Sun Shared Visualization 1.1 software (including the 1.1 Update 1 and 1.1.1 releases) and how the software interacts with other software packages. There is also discussion of supported hardware and Shared Visualization 1.1 server starting techniques. Topics include:

Sun Shared Visualization 1.1 Software Introduction

This document introduces and describes the use of the Sun Shared Visualization 1.1 software advanced visualization technologies, without any instructions on installing or configuring Shared Visualization 1.1 server hosts. That information and system requirements are available in the *Sun Shared Visualization 1.1 Software Server Administration Guide*, 820-3256. See "Related Documentation" on page xvii.

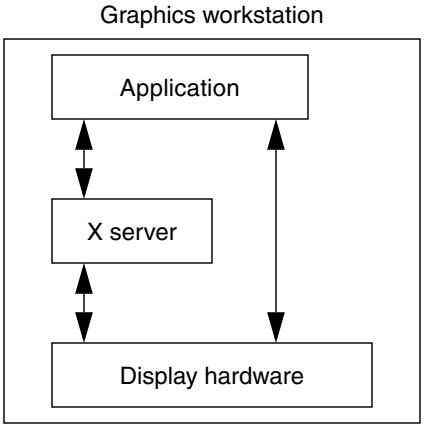
Shared Visualization 1.1 server software enables you to use graphics resources (as well as CPUs, memory, and storage) on the network in place of these resources on your desktop. A graphics server can be in a back room or data center. The graphics server can serve multiple clients serially or simultaneously, aiding in collaboration.

Storage, compute, and graphics processing can be tightly coupled and secure in the server room. The server can have more resources than your desktop, and can yield better performance than running the application on your desktop system.

Traditional Graphics Models

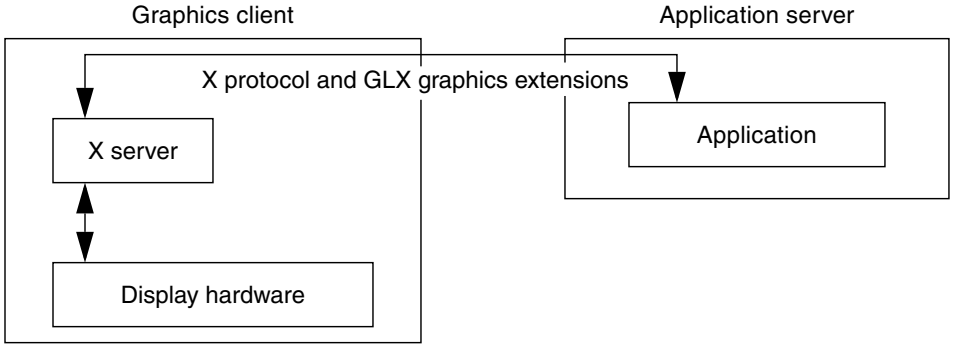
The graphics workstation model in FIGURE 1-1 runs the application on the same host as the user's X server and display hardware. Such desktop systems often lack sufficient resources for demanding applications and large data sets.

FIGURE 1-1 Workstation Graphics



In the application server model in FIGURE 1-2, all graphics pass over the network from the application server to the graphics client.

FIGURE 1-2 Remote X Server Graphics

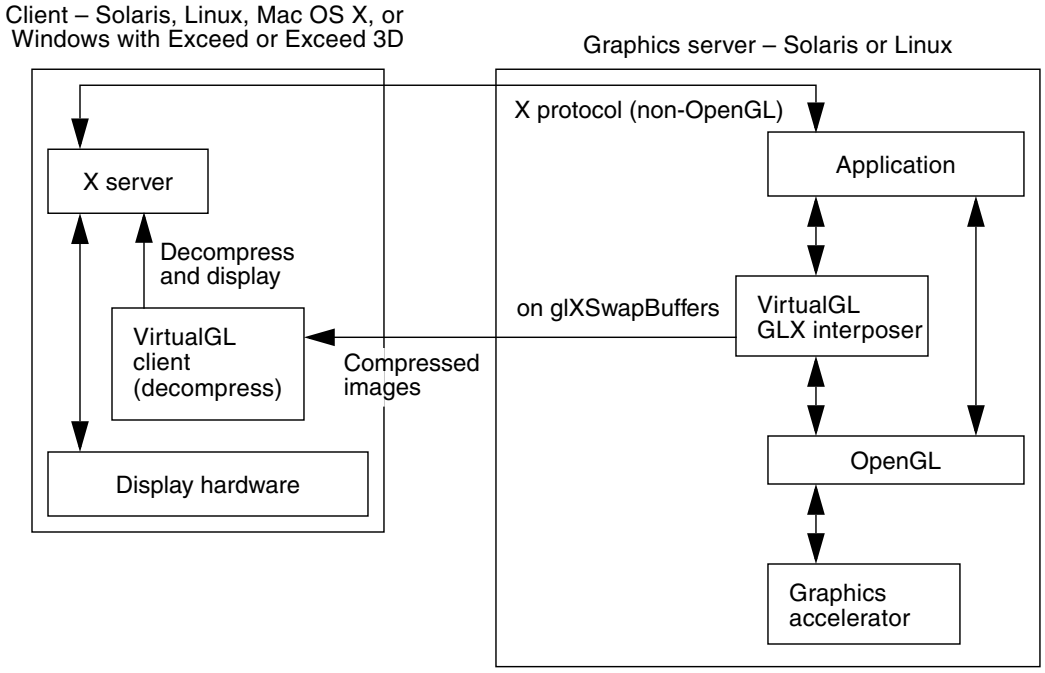


As data sets have increased in size, this data transmission has become more of a burden. Except for performance, an application should run the same whether local to the client or remote.

Sun Shared Visualization 1.1 Model

The Sun Shared Visualization 1.1 model, shown in FIGURE 1-3, runs the application on a server host with sufficient resources, including a graphics accelerator.

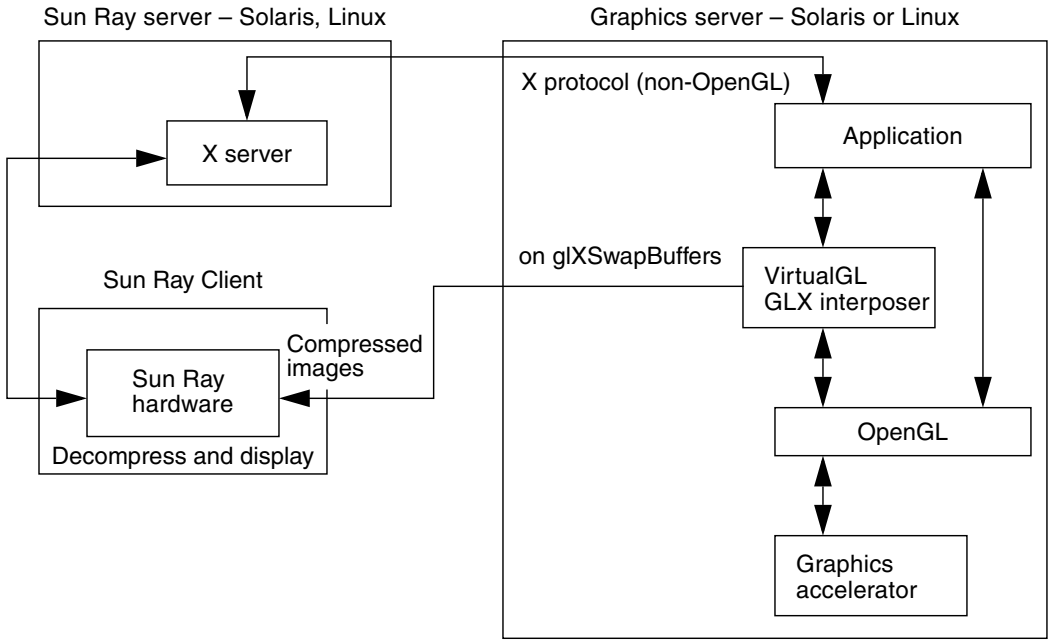
FIGURE 1-3 Shared Visualization 1.1 Server Architecture Using VGL Image Transport



When the application completes drawing an image, the image is read from the graphics hardware, compressed (optional), and sent to the client. The client decompresses the image and displays the pixels for the user. This process is VGL Image Transport. The application's X interactions (mouse and keyboard events, and menu selections) go to the client X server.

The method is similar when the client is a Sun Ray™ thin client, except that the Sun Ray desktop unit (DTU) client hardware decompresses the images. This process is Sun Ray Image Transport. A Sun Ray thin client (DTU) has the keyboard, mouse, and display, but the Sun Ray server runs the client's X server. See FIGURE 1-4.

FIGURE 1-4 Shared Visualization 1.1 Server Architecture Using Sun Ray Image Transport



Software Components

Sun Grid Engine

Sun Grid Engine performs resource management and has been extended for graphics servers to allocate graphics resources, as well as CPUs, memory, and other components. In an environment that has multiple execution servers or multiple graphics accelerators on a host, Sun Grid Engine can select a suitable, lightly-loaded server to run your application, and select a lightly-loaded graphics device on that server. Grid Engine also starts applications on that execution server, so you need not log in to the server.

Job scripts can specify options to Sun Grid Engine. In an environment with heterogeneous execution servers, these options could specify which processor types and operating systems are capable of running the application.

Sun Grid Engine Advance Reservation Server

Advance Reservation (AR) is a feature of some queuing software systems but not yet present in Sun Grid Engine (SGE) release 6.1. (If you are using a later release of Sun Grid Engine, check whether that version includes an Advance Reservation feature.)

The AR requirement is to schedule compute and visualization resources at a time when the computer resources and the persons to use the resources are both available. The Advance Reservation server makes this possible.

If your Sun Grid Engine installation is running the optional AR server, you can request a reservation using a command-line utility or a simple graphical user interface. See "Advance Reservations" on page 73 for more information.

VirtualGL

The key component for remote visualization is VirtualGL (VGL), as shown in FIGURE 1-3 and FIGURE 1-4. VirtualGL interposes on the application, which enables the application to remotely send the graphics transparently (to the application). That is, an unchanged application that was written for a graphics workstation can run on the Sun Shared Visualization 1.1 server and still provide VirtualGL's graphics images to the client's desktop. VirtualGL reads images from the graphics device, compresses the images, and transmits the images to the VirtualGL client software (or to the Sun Ray DTU client hardware).

The compressed images transmitted from the Sun Shared Visualization 1.1 server to the client often require less network bandwidth than transmitting the graphical data (as in the Remote X server model, shown in FIGURE 1-2), and can achieve interactive performance that is comparable or even better.

Advantages of VGL Image Transport compared to TurboVNC, which is introduced in "TurboVNC" on page 7:

- Seamless windows – every application window appears as a separate window on the user's desktop.
- Supports advanced display features, such as quad-buffered stereographic and transparent overlay rendering, if they are available on the client host's X server.
- Offers optional built-in encryption.
- Consumes fewer server CPU cycles, since 2D X11 rendering occurs on the client.

Disadvantages of VGL Image Transport compared to TurboVNC:

- VGL Image Transport does not work well on high-latency networks.
- No collaboration features.
- Requires Exceed for use with Windows clients.

- The client is not stateless. As with any remote X11 application, if the network connection drops, then the application will exit.

TurboVNC

Shared Visualization 1.1 server also includes Virtual Network Computing (VNC) software with optimized compression, called TurboVNC. TurboVNC is suitable for displaying to remote clients on a slow or high-latency network (for example, the Internet), as well as on LANs. VirtualGL reads back images from the graphics accelerator but passes the images uncompressed to TurboVNC's proxy X server (called the TurboVNC server session process) on the graphics server host. VirtualGL uses the X11 Image Transport. This TurboVNC session compresses the images for viewing by one or more remote TurboVNC clients.

Advantages of TurboVNC compared to VGL Image Transport:

- TurboVNC performs very well on low-bandwidth, high-latency connections (such as broadband or long-haul T1 lines). The 3D application's GUI will load and render much faster with TurboVNC than with the VGL Image Transport on such connections.

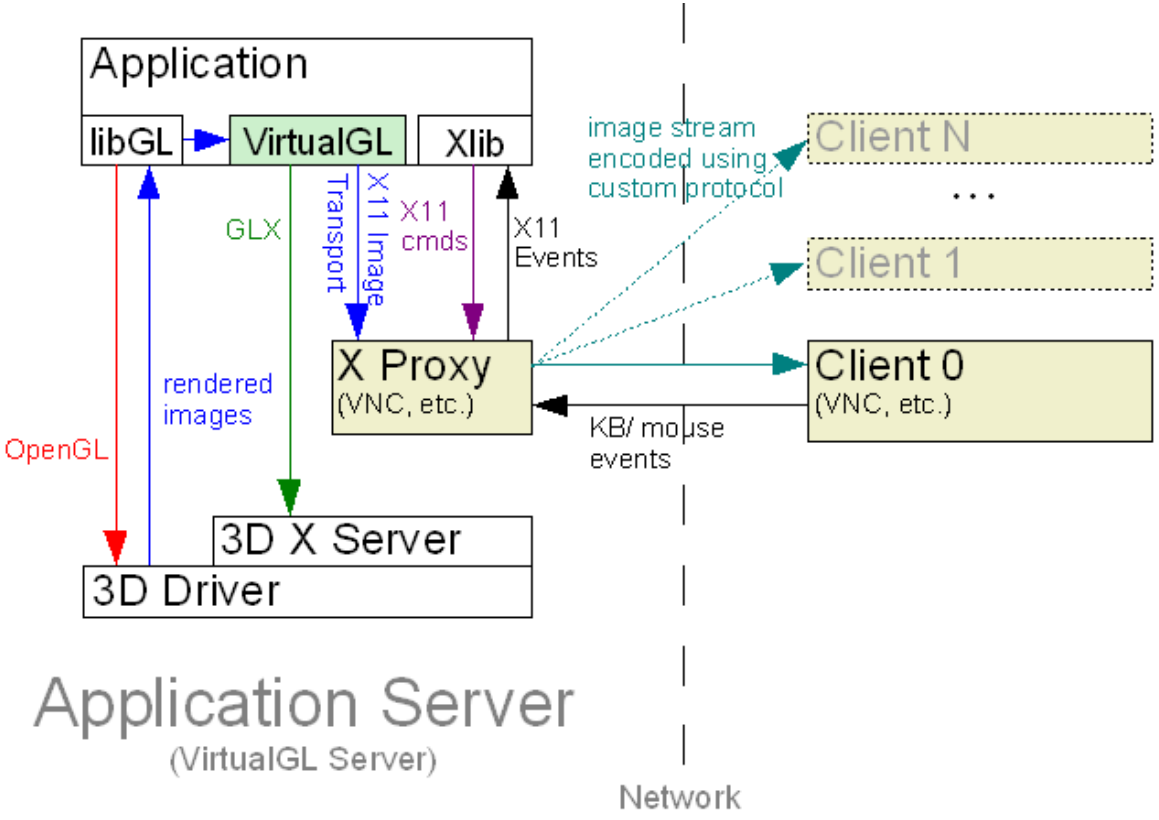
  TurboVNC can also hide network latency by decompressing and drawing a frame on the client while the next frame is being fetched from the server. This feature can improve performance dramatically on high-latency connections.
- TurboVNC provides rudimentary collaboration capabilities. Multiple TurboVNC clients can share viewing of and even interaction with the running programs, passing around control of the keyboard and mouse.
- The TurboVNC client is stateless. If the network hiccups or the client is otherwise disconnected, the session remains running on the server and can be rejoined from any machine on the network.
- No X server is required on the client machine. This situation reduces the deployment cost and complexity for Windows clients.
- For 3D applications whose rendered images do not contain very many unique colors, for instance, design applications in wireframe mode, the hybrid protocol used by TurboVNC 0.5 generally uses less network bandwidth than the pure motion JPEG protocol used by the VGL Image Transport.

Disadvantages of TurboVNC compared to VGL Image Transport:

- No seamless windows. All application windows are constrained to a *virtual desktop*, which displays in a single window on the client machine.
- TurboVNC generally requires about 20% more server CPU cycles to maintain the same frame rate as the VGL Image Transport, both because TurboVNC has to compress more pixels in each frame (an entire desktop rather than a single window) and because TurboVNC has to perform 2D (X11) rendering as well as 3D rendering.

- TurboVNC does not support quad-buffered stereographic or overlay rendering.

FIGURE 1-5 TurboVNC Server Session and Clients – VirtualGL Uses X11 Image Transport



TurboVNC X Extensions

The 3D X Server shown in FIGURE 1-5 is used only to access the 3D graphics accelerator. The application's 2D X server is the proxy implemented by TurboVNC. It supports fewer X extensions than are available on most X servers. The X extensions available within the TurboVNC session are independent of the clients on which `vncviewer` or WebVNC are running.

Note – The application has access to the GLX extension through VirtualGL, even though `xdpyinfo` does not report this.

`xdpyinfo` displays the following X extensions as being supported by TurboVNC:

- BIG-REQUESTS
- MIT-SHM
- MIT-SUNDRY-NONSTANDARD
- SHAPE
- SYNC
- XC-MISC
- XTEST

For instructions on using TurboVNC, refer to the TurboVNC man pages using the following command:

```
man -M /opt/TurboVNC/man {vncserver | Xvnc | vncviewer | vncconnect | vncpasswd}
```

Note – For Windows, use the embedded help feature (question mark in upper-right corner of the window).

Supported Platforms

Server Platforms

TABLE 1-1 describes the server platforms supported by the Sun Shared Visualization 1.1 software.

TABLE 1-1 Supported Server Platforms*

Processor Architecture	Operating System	OS Releases
UltraSPARC®	Solaris OS	At least Solaris 8, and at least **OpenSolaris™ release 2008.11**
x86	Solaris OS	At least Solaris 10, and at least **OpenSolaris™ release 2008.11**
x86	Linux	Red Hat Enterprise Linux 3, 4, and 5; SuSE 9 and 10, **Ubuntu 8**

* **Boldface** platforms are new in the 1.1.1 release.

To use the optional Advance Reservation facility, the server (or client) requires a Java™ runtime environment (JRE). The earliest version to support Advance Reservation is JRE 1.5 (known as Java 5).

Server Graphics Accelerators

TABLE 1-2 describes the graphics accelerators supported by the Sun Shared Visualization 1.1 software, for respective processor architectures.

TABLE 1-2 Server Graphics Accelerators

Processor Architecture	Graphics Accelerators	Comments
UltraSPARC	XVR-2500	Suitable for stereographic display
	XVR-1200	Not suitable for stereographic display
	XVR-600	Not suitable for stereographic display
x86	NVidia Quadro series	
	NVidia Quadro Plex series	

The Sun Shared Visualization 1.1 software also supports Chromium clusters, when the Chromium Head Node is configured as a graphics server.

Client Platforms

TABLE 1-3 describes the client platforms supported by the Sun Shared Visualization 1.1 software.

TABLE 1-3 Supported Client Platforms*

Processor Architecture	Minimum CPU Clock Speed	Operating System	OS Releases
UltraSPARC	900MHz	Solaris OS	At least Solaris 8, and at least **OpenSolaris™ release 2008.11**
x86	1.0 GHz	Solaris OS	At least Solaris 10, and at least **OpenSolaris™ release 2008.11**

TABLE 1-3 Supported Client Platforms* *(Continued)*

Processor Architecture	Minimum CPU Clock Speed	Operating System	OS Releases
x86	1.0 GHz	Linux	RedHat Enterprise Linux 3, 4, and 5; SuSE 9 and 10, **Ubuntu 8**
x86	1.0 GHz	Windows	Windows XP or Vista. VGL Image Transport requires Exceed 2006 or later (or Exceed 3D or later for stereographic display support)
x86-based Macintosh	1.0 GHz	Mac OS X	Mac OS X 10.4 (Tiger) and 10.5 (Leopard)

* **Boldface** platforms are new in the 1.1.1 release.

Minimally, the client must:

- Support 24- or 32-bit pixel true color display
- For stereographic display support or to use transparent overlays, the client must also have a high-end 3D graphics accelerator installed.

Note – If the client host is using a 3D graphics accelerator, install the vendor's current OpenGL® library and drivers for that 3D accelerator.

To use the optional Advance Reservation facility, the client requires a Java runtime environment (JRE). The earliest version to support Advance Reservation is JRE 1.5 (known as *Java 5*).

Shared Visualization 1.1 Server Starting Techniques

Startup Methods

Chapter 3 and Chapter 4 give alternative ways to start the Sun Shared Visualization 1.1 server:

- Chapter 3 – Manual starting

- Chapter 4 – Using Sun Grid Engine, through job scripts or with options on the command line

These methods are ordered from the simplest to understand to the more complex. However, you might find Sun Grid Engine job scripts the easiest to use. The scripts reduce typing and repetition, because the job script passes specifications to Sun Grid Engine for you.

Client Types

Depending on your situation, you might have a choice among Shared Visualization 1.1 server clients:

- Sun Ray thin client, using the Sun Ray's hardware image decompression.

  This client uses VirtualGL's Sun Ray plug-in (Sun Ray Image Transport) and offers a seamless window experience.

- VirtualGL client software (`vglclient`) on a UNIX (Solaris or Linux) host.

  This software uses VGL Image Transport (formerly called Direct mode) to provide a seamless window experience and good performance on a LAN.

- VirtualGL client software (`vglclient`) on a Mac OS X host.

  Use of `vglclient` software on a Mac OS X host is nearly identical to use of `vglclient` on a UNIX host, except that you must explicitly start the X server on the Mac OS X host.

- VirtualGL client on Window PCs using the Exceed 2006 (or newer) X server.

  This client also uses VGL Image Transport for a seamless window experience.

Note – For applications that use stereographic or transparent overlays, Exceed 3D is required on a Windows client.

- TurboVNC session.

  This option performs best over a wide-area network (WAN) and offers multiple client collaboration. Each client of the TurboVNC session needs a TurboVNC viewer:

  - The Java based WebVNC viewer for use in a web browser (simple)
  - A dedicated `vncviewer` client software component (better-performing)

  This uses VirtualGL in X11 Image Transport (formerly called Raw mode). Uncompressed images are given to the TurboVNC server session (X Proxy) to compress. The TurboVNC session is within one window on the client desktop.

Client Software Installation Matrix

Chapter 2 describes installation of the Shared Visualization 1.1 software. TABLE 1-4 directs you to the appropriate installation instructions (if any) for your client type.

TABLE 1-4 Client Software Installation Matrix

Client Type	VirtualGL Image Transport Used	Other Characteristic	Installation
Sun Ray client	Sun Ray Image Transport to Sun Ray thin client		No installation is necessary – neither on Sun Ray nor on client's Sun Ray server
VirtualGL client	VGL Image Transport to client	UNIX VirtualGL client	See "Installation on a Solaris or Linux Client" on page 16
		Mac OS X VirtualGL client	See "Installation on a Mac OS X Client" on page 23
		Windows VirtualGL client	See "Installation on a Windows Client" on page 26
TurboVNC client	X11 Image Transport to TurboVNC server session, VNC transport to client	UNIX TurboVNC viewer	See "Installation on a Solaris or Linux Client" on page 16
		Mac OS X TurboVNC viewer	See "Installation on a Mac OS X Client" on page 23
		Windows TurboVNC viewer	See "To Install TurboVNC on a Windows Client" on page 26
		Java based TurboVNC web browser applet	No installation is necessary

Startup Method Guide

TABLE 1-4 provides a guide to the starting methods for each client type. Each cell of the matrix provides a link to the section describing that startup method.

TABLE 1-5 Startup Methods for Each Client Type

Client Type	Manually Starting	Submitting a Sun Grid Engine Job
Sun Ray client	"Using VirtualGL From a Sun Ray Client" on page 34	Chapter 4, "Using Sun Grid Engine to Start the Sun Shared Visualization 1.1 Software" on page 57
UNIX or Mac OS X VirtualGL client	"Using VirtualGL From Other Clients" on page 35	Chapter 4, "Using Sun Grid Engine to Start the Sun Shared Visualization 1.1 Software" on page 57
Windows VirtualGL client	"Using VirtualGL From a Windows Client" on page 37	Chapter 4, "Using Sun Grid Engine to Start the Sun Shared Visualization 1.1 Software" on page 57
VNC viewer or web browser	"Manually Using TurboVNC" on page 42	"Submitting Sun Grid Engine TurboVNC Jobs" on page 67

CHAPTER 2

Sun Shared Visualization 1.1 Client Installation

This chapter describes installation and configuration information for the Sun Shared Visualization 1.1 client. Topics include:

Note – Unless stated otherwise, the majority of examples provided in this chapter are for the Solaris 10 Operating System.

Note – Names of files and directories that indicate a software release name might be slightly different from the names specified in the Sun Shared Visualization 1.1 documentation. For example, instead of "1.1" you might find "1.1.1".

Sun Shared Visualization 1.1 Software

This section describes how to install the Sun Shared Visualization 1.1 software onto your client and how to remove the software.

- Installation on a Solaris or Linux client installs support for both VirtualGL and TurboVNC.
- Installation on a Mac OS X client installs either VirtualGL or TurboVNC, or both.
- Installation on a Windows client installs either VirtualGL or TurboVNC, or both.
  - A Windows client using VirtualGL Image Transport requires both VirtualGL and the third-party Exceed (or Exceed 3D) X windows server. (Exceed software is not included with Sun Shared Visualization software).
  - Using TurboVNC does not require Exceed software.
- A Sun Ray client has nothing to install. (The graphics server has Sun Shared Visualization software installed.)

Installation on a Solaris or Linux Client

Software Components That Are Not Needed on a Client

If you are using Sun Grid Engine, your client mounts that software from your grid's NFS server. You do not need to install a copy on the client. Therefore, the product's optional software (listed in TABLE 2-1) is not required on the client.

TABLE 2-1 Optional Software Components, All Unneeded on Clients

Solaris packages	Linux RPMs
`SUNWsge3D`	`sun-n1ge-3D.noarch.rpm`

`SUNWsgear`
`SUNWsgeau`
`SUNWsgearsmr`
	`sun-n1ge-adv_reserv.noarch.rpm`

The Sun Ray plug-in (Solaris package `SUNWvglsr` or Linux RPM `VirtualGL-SunRay`) is installed by default. Though not harmful, this plug-in is not needed on any clients. For instructions to remove the plug-in after installation, see "To Remove the Sun Ray Plug-In" on page 21.

▼ To Install Sun Shared Visualization 1.1 Software on a Solaris or Linux Client

1. **Take one of the following actions, depending on your installation media:**
   - Perform Step 2 on page 17 if you are installing the software from a download directory.
   - Perform Step 3 on page 17 if you are installing the software from the CD-ROM.
2. **Install the software from a download directory.**
   a. **As superuser, change to that directory and extract the zip file.**

```
# cd /path/to/download/directory
# unzip SharedVisualization_1.1_package.zip
```

   where `1.1` is `1.1.1` in the 1.1.1 release, and *package* depends on the download, `solaris` or `linux`. The directory structure is created and the files are extracted.

   b. **Change to the installation script directory:**

```
# cd SharedVisualization_1.1_package
```

   c. **Continue to** Step 4 on page 19.
3. **Install the software from the CD-ROM.**
   a. **As superuser, insert the Sun Shared Visualization 1.1 CD-ROM into an optical drive that is connected to your system.**

   If your system automatically mounts the disc, continue to Step b on page 18.

   If your system does not automatically mount the disk, mount it with the following commands:

```
# mkdir -p /cdrom/SSV1.1
# mount -F hsfs -o ro device /cdrom/SSV1.1
```

   where `1.1` is `1.1.1` in the 1.1.1 release, and *device* is:
   - a path (such as `/dev/dsk/c0t6d0s2` for the Solaris OS) that is obtained by running the `rmformat` command, using `dsk` rather than `rdsk`

- the path `/dev/cdrom` for Linux

If you are installing the Sun Shared Visualization 1.1 software from a CD-ROM onto a Linux host, you might see the following error:

```
bash: ./install: /bin/bash: bad interpreter: Permission denied
```

This error might occur if you use the automounter with default options, or if you have `noexec` in the CD-ROM mount entry of the `/etc/fstab` file.

To prevent this error, change the `noexec` option to `exec`, or mount the CD-ROM manually using the `exec` option.

b. Change to the installation directory:

```
# cd cdrom-path
```

where the *cdrom-path* depends on your environment. The following are common names that are available in each environment, but the name on your system might vary:

Environment	CD-ROM Path
Solaris OS	`/cdrom/ssv_1.1`
Red Hat Linux	`/cdrom/ssv_1.1` or `/cdrom/SSV_1.1` or `/media/cdrom`
SuSE Linux	`/media/SSV_1_1` or `/media/dvd`
Ubuntu Linux	`/cdrom/` or `/media/cdrom` or `/media/cdrom0`

If you used the `mkdir` command in Step a on page 17 in any environment, you can change to the installation directory with this command:

```
# cd /cdrom/SSV1.1*
```

c. Continue to Step 4 on page 19.

4. **Run the installation script:**

```
# ./install
```

The script begins:

```
Sun Microsystems, Inc. ("Sun") ENTITLEMENT for SOFTWARE

Licensee/Company: Entity receiving Software.

Effective Date: Date of delivery of the Software to You.
....
```

The script displays the licensing agreement, and asks:

```
...
Agreement. No modification of this Agreement will be binding, unless in writing
and signed by an authorized representative of each party.

Please contact Sun Microsystems, Inc. 4150 Network Circle, Santa Clara,
California 95054 if you have questions.


Do you accept the license agreement? [y/n]:
```

5. **To proceed with software installation, type** `y`.

After agreement, the script begins installation:

```
This program installs the software for the Sun Shared Visualization 1.1

Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
```

The script checks for a newer version of the Sun Shared Visualization 1.1 software. If the script finds one, the script displays:

```
This system has a higher version of Sun Shared Visualization
software than is available in this Release.  Sun Shared
Visualization software from this release will not be installed.
```

Otherwise, the script begins adding packages and asks you about optional software:

```
application SUNWsge3D    Sun Grid Engine Graphic Extensions
application SUNWsgearsmr Sun Grid Engine Graphic Advance Reservations
application SUNWsgeau    Sun Grid Engine Graphic Advance Reservations (Usr)

Do you wish to install the optional Software (SUNWsge3D SUNWsgeau
SUNWsgearsmr)?  [y,n,?,q]
```

The details of this question will vary, depending on your operating system.

6. **Answer** `n`.

   The optional software is needed only on servers. See "Software Components That Are Not Needed on a Client" on page 16. The script informs you:

```
This script is about to take the following actions:
- Install Sun Shared Visualization Software

To cancel installation of this software, press 'q' followed by a Return.
       **OR**
Press Return key to begin installation:
```

7. **Press the Return key to continue installation.**

   The script begins installing required patches and packages:

```
*** Installing Sun Shared Visualization Software for Solaris 10...
Installing required packages:
       SUNWtvnc SUNWvgl SUNWvglsr SUNWvrpt

Installation of <SUNWtvnc> was successful.
Installation of <SUNWvgl> was successful.
Installation of <SUNWvglsr> was successful.
Installation of <SUNWvrpt> was successful.

*** Installation complete.
```

The script informs you how to remove the software, and where a log file of the installation is located:

```
To remove this software, use the 'remove' script on this CDROM, or
the following script:

       /var/tmp/SharedVis_remove

A log of this installation can be found at:
       /var/tmp/SharedVis.install.2007.12.22.0952
```

The log file is named with a date and time stamp. In this example, December 22, 2007 at 9:52 am.

Secure Shell on Solaris 8 and 9

`vglconnect` uses the secure shell (ssh), which may not be installed on older Solaris releases. If your need to use Sun Shared Visualization on a Solaris host without ssh, you need to install ssh, for example, from
http://www.sunfreeware.com/openssh8.html or
http://www.sunfreeware.com/openssh9.html

▼ To Remove the Sun Ray Plug-In

The Solaris package `SUNWvglsr` and the Linux RPM `VirtualGL-SunRay` are server support for Sun Ray clients. This plug-in is not needed on a client, though it is installed by the installation script on a Solaris or Linux host. See "Software Components That Are Not Needed on a Client" on page 16. When the installation script is completed, you can leave the plug-in installed or you can remove it.

- **Remove the Sun Ray plug-in.**
  - For Solaris clients, type:

```
# pkgrm SUNWvglsr
```

  - For Linux clients, type:

```
# rpm -e VirtualGL-SunRay
```

▼ To Remove the Sun Shared Visualization 1.1 Software From Solaris or Linux Clients

You might need to remove the Sun Shared Visualization 1.1 software in the future.

1. **As superuser, run the removal script.**
   - If you are running the removal script installed with the Sun Shared Visualization 1.1 software, type:

```
# /var/tmp/SharedVis_remove
```

   - If you are running the removal script from the CD-ROM:

   a. Mount the CD-ROM as in Step a on page 17

   b. Then `cd` onto the CD-ROM as in Step b on page 17**.**

   c. Then type one of the following commands:
      - For Solaris OS:

```
# SharedVisualization_1.1/Solaris/remove
```

      - For Linux:

```
# SharedVisualization_1.1/Linux/remove
```

   The script starts and identifies the software packages that are to be removed.

```
All required software for the Sun Shared Visualization Software
software  will be REMOVED.

The following packages will be removed:
 SUNWvglsr SUNWvgl SUNWtvnc SUNWvrpt
```

   The script asks:

```
To cancel removal of this software, press 'q' followed by a Return.
       **OR**
Press Return key to begin package removal:
```

2. **Press the Return key to begin package removal.**

   Pressing the Q key and the Return key aborts the script.

   The script does a search for the installed packages and displays the progress.

```
*** Found the following packages to remove:
            SUNWvglsr SUNWvgl SUNWtvnc SUNWvrpt
*** Removing old package(s)...

Removal of <SUNWvglsr> was successful.

Removal of <SUNWvgl> was successful.

Removal of <SUNWtvnc> was successful.

Removal of <SUNWvrpt> was successful.
```

The script concludes and tells you where a log file of the removal is located.

```
*** Done.  A log of this removal can be found at:
      /var/tmp/SharedVis.remove.2007.12.22
```

The log file is named with a date stamp. In this example, December 22, 2007.

Installation on a Mac OS X Client

As explained in Chapter 1 (especially through FIGURE 1-3 and FIGURE 1-5), Sun Shared Visualization software offers two alternative image transport techniques, VirtualGL Image Transport and TurboVNC (using X11 Image Transport). On a Macintosh client, you can install software for either or both forms of image transport.

The Sun Shared Visualization 1.1 CD-ROM did not include the Macintosh installation files (with names including the `.dmg` extension), but the 1.1.1 release does. This installation software is also available for download. Software for VirtualGL and TurboVNC are downloaded together. You can choose to install either or both.

▼ To Download Sun Shared Visualization 1.1 Client Software for Mac OS X

1. **Select the software download option starting from this page:**

   http://www.sun.com/servers/cr/visualization/get_it.jsp

   Follow instructions at this site for downloading Sun Shared Visualization software and documentation.

2. **Open the Mac OS X download file.**

   If the file does not unzip automatially, double-click on the `.zip` file.

   Your desktop should now have two `.dmg` files:

   - `TurboVNC-`*version*`.dmg`
   - `VirtualGL-`*version*`.dmg`

   The *version* number indicates the release of TurboVNC or VirtualGL software.

▼ To Install VirtualGL on a Mac OS X Client

Installing the VirtualGL package on an x86 Apple Macintosh system enables that machine to act as a VGL Image Transport client.

1. **Install the X11 application if it isn't already present on the Macintosh system.**

   X11 is available on the Mac OS X installation media in the Optional Installs package.

2. **Uninstall** `Applications/VirtualGL` **if a different version is already present on the Macintosh system.**

3. **On the Mac OS X desktop, double-click the VirtualGL disk image.**

   The VirtualGL disk image is named `VirtualGL-`*version*`.dmg`.

4. **Open** `VirtualGL-`*version*`.pkg` **inside the disk image.**

   Follow any on-screen instructions to install the Mac OS X client. This client-only Mac OS X package will install files in the same locations as files installed by the Linux RPM.

▼ To Install TurboVNC on a Mac OS X Client

Installing the TurboVNC package on an x86 Apple Macintosh system enables that system to start an optomized TurboVNC viewer. VirtualGL can be installed on the same system, but this viewer doesn't make use of the VirtualGL Mac OS X client. Instead, the graphics server will run VirtualGL and use X11 Image Transport to

provide images to the TurboVNC server session (X proxy). The TurboVNC session transports the images to one or more clients, which display the images with the TurboVNC verwer or the Web VNC viewer.

1. **Uninstall `Application/TurboVNC` if a different version is already present on the Macintosh system.**
2. **On the Mac OS X desktop, double-click the TurboVNC disk image.**

   The TurboVNC disk image is named `TurboVNC-`*version*`.dmg`.
3. **Open `TurboVNC-`*version*`.pkg` inside the disk image.**

   Follow any on-screen instructions to install the Mac OS X client. This client-only Mac OS X package will install files in the same locations as files installed by the Linux RPM.

Removing Sun Shared Visualization 1.1 Software From a Mac OS X Client

You might need to remove the Sun Shared Visualization 1.1 software in the future.

▼ To Remove Sun Shared Visualization 1.1 Software From a Mac OS X Client

If your Macintosh system already has a package removal function, use that function to remove the Sun Shared Visualization 1.1 software in the future. Otherwise, perform this procedure.

1. **Download and install the latest version of OSXPM from `http://www.osxgnu.org`.**
2. **Start OSXPM.**
3. **Click the Delete Package button.**
4. **Find and highlight any of these package names in the list of packages:**
   - `TurboVNC-`*version*`.pkg`
   - `VirtualGL-`*version*`.pkg`
5. **Click Delete Selected.**

   Enter your password, if prompted to do so.

Installation on a Windows Client

As explained in Chapter 1 (especially through FIGURE 1-3 and FIGURE 1-5), Sun Shared Visualization software offers two alternative image transport techniques, VirtualGL Image Transport and TurboVNC (using X11 Image Transport):

- Using VirtualGL Image Transport on a Windows client requires installing both VirtualGL and the third-party Exceed (or Exceed 3D) X windows server (which is not included with Sun Shared Visualization software). VirtualGL provides a more seamless window experience.
- Using TurboVNC requires installing only TurboVNC. This technique provides a single window to each TurboVNC session (on a remote graphics server). TurboVNC also allows collaborative sharing of a TurboVNC session.

Determine if you need to obtain Exceed and install VirtualGL, install TurboVNC, or both.

The self-expanding `.exe` installers are on the CD-ROM in the `SharedVisualization_1.1*/Windows` directory. (The same directory is available after unzipping the `SharedVisualization_1.1*_windows.zip` download file.)

▼ To Install TurboVNC on a Windows Client

Installing the TurboVNC package on a Windows machine enables the machine to start a TurboVNC viewer. You do *not* need to install VirtualGL on the Windows client. VirtualGL will run on the graphics server, and will use its X11 Image Transport to provide images to the TurboVNC server session (X proxy). TurboVNC transports the images to one or more clients, to be displayed by the TurboVNC viewer or the Web VNC viewer.

1. **Locate the TurboVNC installer, `TurboVNC.exe`.**
2. **Run the TurboVNC installer, typically by double-clicking its icon.**

   The only configuration option for installation is the directory in which you want the files to be installed.

Enabling VirtualGL Image Transport on a Windows Client

Installing the VirtualGL package on a Windows machine enables the machine to act as a VGL Image Transport client, along with Exceed or Exceed 3D. You do *not* need to install VirtualGL or Exceed on the Windows client machine if only TurboVNC clients will be used.

▼ To Install VirtualGL on a Windows Client

1. **Locate the VirtualGL installer,** `VirtualGL.exe`**.**
2. **Run the VirtualGL installer, typically by double-clicking its icon.**

   The only configuration option for installation is the directory in which you want the files to be installed.

▼ To Install Exceed for Windows

1. **Install Exceed or Exceed 3D, if the software isn't already installed.**

   See the Exceed documentation for instructions.

2. **Install patches for Exceed or Exceed 3D, if the patches aren't already installed.**

   If your Windows client is using Exceed 2008 to support VirtualGL, you can optimize performance (as much as a 20 percent gain) by enabling the MIT Shared Memory Extension (MIT-SHM extension).

   Obtain and install the latest Exceed 2008 Service Pack, including `xlib` patch `xlib.dll` v13.0.1.235 (or higher) for Exceed 2008. Patches are available to those with a Hummingbird support account from the Hummingbird support site:

```
http://connectivity.hummingbird.com/support/nc/exceed_patches.html
```

   The *Sun Shared Visualization 1.1.1 Release Notes* might contain more recent information about patches for this product.

3. **Add the Exceed software path to your system** `PATH` **environment variable.**

   See the Exceed documentation for instructions, if its path has not already been added.

▼ To Disable Pixel Format Conversion (for Exceed 2006 and Earlier)

1. **Load Exceed XConfig.**

   a. **Right-click on the Exceed taskbar icon.**

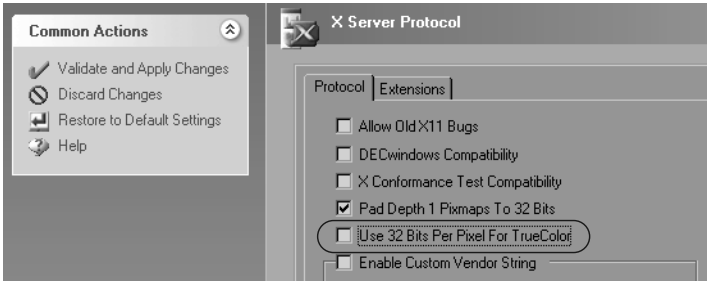
   b. **Select Tools, then Configuration.**

2. **Open the X Server Protocol applet in XConfig.**

Note – If you are using the Classic View mode of XConfig, open the Protocol applet instead.

---

3. **In the X Server Protocol applet, select the Protocol tab and ensure that the Use 32 Bits Per Pixel For True Color box is unchecked.**



4. **Click Validate and Apply Changes.**

Note – If XConfig asks whether you want to perform a server reset, click Yes.

---

5. **Proceed to** "To Disable the Backing Store" on page 28**.**

▼ To Disable the Backing Store

1. **Load Exceed XConfig.**

   a. **Right-click on the Exceed taskbar icon.**

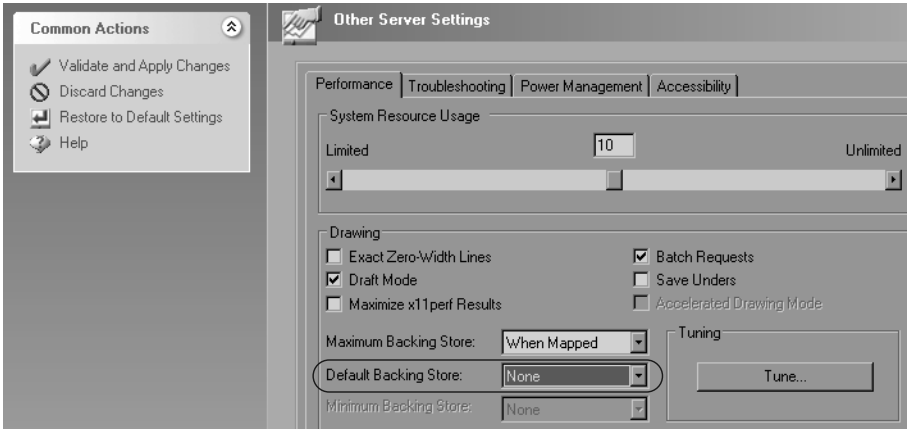
   b. **Select Tools, then Configuration.**

2. **Open the Other Server Settings applet in XConfig.**

Note – If you are using the Classic View mode of XConfig, open the Performance applet instead.

3. **Select the Performance tab and ensure that Default Backing Store is set to None.**

.



4. **Click Validate and Apply Changes.**

Note – If XConfig asks whether you want to perform a server reset, click Yes.

▼ To Obtain Optimal Performance With Exceed

VirtualGL can use the MIT-SHM extension in the Exceed software to accelerate image drawing on Windows clients. By using this extension, the overall performance of the VirtualGL pipeline can be improved by as much as 20%. However, Exceed 2008 requires a patch to enable its MIT-SHM extension to work with VirtualGL. See the *Sun Shared Visualization 1.1.1 Release Notes* for details.

1. **Load Exceed XConfig.**

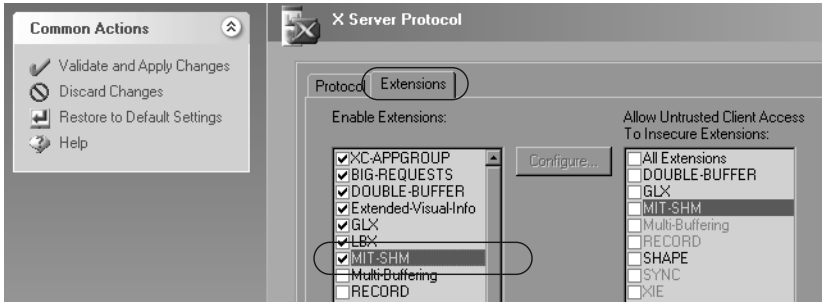
   a. **Right-click on the Exceed taskbar icon.**

   b. **Select Tools, then Configuration.**

2. **Open the X Server Protocol applet in XConfig.**

Note – If you are using the Classic View mode of XConfig, open the Protocol applet instead.

3. **In the X Server Protocol applet, select the Extensions tab and ensure that MIT-SHM is checked.**



4. **Click Validate and Apply Changes.**

Note – If XConfig asks whether you want to perform a server reset, click Yes.

Removing Sun Shared Visualization 1.1 Software From a Windows Client

You might need to remove the Sun Shared Visualization 1.1 software in the future.

▼ To Remove the Sun Shared Visualization 1.1 Software From a Windows Client

- **Do one of the following:**
  - In a Windows XP client, use the Add or Remove Programs applet in the control panel.
  - In a Windows Vista client, use the Programs and Features applet.

CHAPTER 3

Manually Using the Sun Shared Visualization 1.1 Software

Topics discussed in this chapter include:

Manual Startup Overview

If you know which host is your graphics server, you can start the Sun Shared Visualization 1.1 server manually. You will either need to know which graphics device or X server on that host will be used, or you will use the server's default.

Note – VirtualGL's default is used if neither the `vglrun -d` option is used nor the `VGL_DISPLAY` environment variable is set on the graphics server when vglrun is invoked. For more on `VGL_DISPLAY`, see "VirtualGL Options and Environment Variables" on page 86.

The procedures in this chapter assume that the graphics server already has Sun Shared Visualization 1.1 software installed and configured as described in Chapter 3 and Chapter 4 of the *Sun Shared Visualization 1.1 Server Administration Guide*. When the server is configured that way, access to the graphics accelerator device (and to

the server's X server, if necessary) is granted either to all users or to the `vglusers` group. In the latter case, you need to verify that the administrator has added your login to that group.

Instructions for use of VirtualGL can vary, depending on:

- Your desired Image Transport (the alternatives are described in Chapter 1).
- Your client type (Sun Ray thin clients, Solaris and Linux UNIX clients – including Mac OS X clients, and Windows clients). For more information, see "Client Types" on page 11.
- Whether you are using TurboVNC. For more information, see "Client Types" on page 11.

In this chapter, use of VirtualGL's Sun Ray Image Transport and VGL Image Transport is described first, followed by information on using TurboVNC.

While this chapter describes manual starting of the Sun Shared Visualization 1.1 software, self-selection of a graphics device in a shared environment is not advised, as other users might be using or about to use that device. If you select a device that others are using, any process sharing the device could exhaust resources (for example, memory on the graphics accelerator). This exhaustion would cause that process to quit or all processes sharing the device to become unreasonably slow. Therefore, sites might prefer to let Sun Grid Engine perform the allocation, as described in the *Sun Shared Visualization 1.1 Software Server Administration Guide*, 820-3256. Chapter 4 describes using Shared Visualization software with Sun Grid Engine.

VirtualGL Startup Sequence

There are two components to be started for remote visualization using VirtualGL's VGL or Sun Ray Image Transports:

- VirtualGL runs on the graphics server, and starts the graphics application. `vglrun` interposes between the application and the GLX and OpenGL® libraries, so VirtualGL can read back completed images from the graphics accelerator and pass the images to the client for display.
- The VirtualGL client software runs on the client (host), receiving images from the graphics server and displaying the images. If the client is a Sun Ray, the Sun Ray hardware and firmware performs this action instead.

The following sections describe use of VirtualGL for Sun Ray, UNIX (Solaris, Linux, or Mac OS X) clients, and Windows clients. When the graphics application starts to use OpenGL on the server, VirtualGL connects to the VirtualGL client (or Sun Ray) and starts streaming compressed image sequences to the client. This mode is not recommended for use on low-bandwidth or high-latency networks.

Note – To use TurboVNC, which is better for low-bandwidth or high-latency networks, see "Manually Using TurboVNC" on page 42.

vglrun Syntax Summary

Within a session to the graphics server, you start your application under control of vglrun. The command's syntax is:

```
/opt/VirtualGL/bin/vglrun [vglrun-options] application [application-arguments]
```

On a Linux graphics server, vglrun is also in /usr/bin, which is included in your path. On a Solaris graphics server, you can add /opt/VirtualGL/bin to your path. In these cases, you can type vglrun without specifying its path. The syntax is therefore:

```
vglrun [vglrun-options] application [application-arguments]
```

You can provide vglrun options prior to the name of the graphics application, and options for the application afterward. For example, vglrun options can improve image quality at the expense of performance and network bandwidth. See "VirtualGL Reference" on page 81 for vglrun options.

vglrun Verification

Before attempting a more complex graphics application, you might want to verify that VirtualGL can communicate with your client. To verify communication efficiently, first run a simple application, such as /opt/VirtualGL/bin/glxspheres.

Using VirtualGL From a Sun Ray Client

If your client is a Sun Ray thin client, the VirtualGL Sun Ray plug-in will use the Sun Ray image transport. That is, the plug-in will compress images for Sun Ray and send these images directly to the Sun Ray DTU for Sun Ray client hardware decompression and display. The Sun Ray environment does not use VirtualGL client software (`vglclient`).

There are two cases for using the Sun Shared Visualization 1.1 software from a Sun Ray client, depending on whether the graphics server is also your Sun Ray server.

▼ To Use VirtualGL From a Sun Ray Client When the Sun Ray Server and the Graphics Server Are Different Hosts

1. **Open a new terminal window that will be dedicated to the graphics server session.**
2. **In the same terminal window, open a Secure Shell (SSH®) session into the graphics server with the `ssh` command:**

```
sunrayserver% ssh -X user@graphics-server
```

Replace *user* with your user account name on the graphics server. If your account name is the same on the current host as on the graphics server, then the *user*`@` can be omitted. Replace *graphics-server* with the hostname (or IP address) of your graphics server.

3. **Within the `ssh` session, start a graphics application using `vglrun`:**

```
graphics-server% /opt/VirtualGL/bin/vglrun [vglrun-options] my-program [my-arguments]
[VGL] NOTICE: Automatically setting VGL_CLIENT environment variable to
[VGL]    100.200.30.45, the IP address of your SSh client.
```

The VirtualGL Sun Ray Image Transport will be used. `ssh` will set your `DISPLAY` environment variable for you to the graphics-server end of an X tunnel. The result is that the X command stream is encrypted and routed to your Sun Ray server. However, VirtualGL detects this situation and transmits images directly to your Sun Ray DTU.

▼ To Use VirtualGL From a Sun Ray Client When the Sun Ray Server Is the Graphics Server

- **In any terminal window, start any graphics application using `vglrun`:**

```
graphics_server% /opt/VirtualGL/bin/vglrun [vglrun-options] my-program [my-arguments]
[VGL] NOTICE: Automatically setting VGL_CLIENT environment variable to
[VGL]    100.200.30.45, the IP address of your SSh client.
```

The VirtualGL Sun Ray Image Transport will be used.

Using VirtualGL From Other Clients

This section describes using VirtualGL's VGL Image Transport from Solaris, Linux, Mac OS X, and Windows clients. Use this mode on local-area networks. Procedures in this section assume you are already logged into the client.

This section has slight variations based on your security choice:

- **X11 Forwarding, VGL unencrypted** – The X11 traffic is encrypted, but the VirtualGL image stream is left unencrypted to maximize performance.
- **SSL-Encrypted VGL Images** – Both X11 traffic and the VirtualGL image stream are encrypted. However, enabling SSL (Secure Socket Layer) encryption can reduce VirtualGL performance by as much as 20% on a high-speed network such as fast (100 Mbps) Ethernet.
- **X11 Forwarding, `ssh`-Encrypted VGL** – Both X11 traffic and the VirtualGL image stream are tunneled through the `ssh` connection, providing a secure solution. The design of your network and your security policy might regulate you to use encrypted VGL Image Transport. However, using `ssh` tunneling can reduce VirtualGL performance by 20-40% on a high-speed network such as fast (100 Mbps) Ethernet, especially for Windows clients.

  Also, in this case, `vglconnect` will make two `ssh` connections into the server (the first to find an open port on the server and the second to create the secure image tunnels and open the Secure Shell). If you are not using an `ssh` agent to create password-less logins, then this mode will require you to enter your password twice.

Before each use, start by deciding which security choice you will use.

▼ To Use VirtualGL From a UNIX or Mac OS X Client

1. **Start the client's X server and log into the client.**

   On a Mac OS X client, start the X11 application that was installed in `Applications/Utilities/X11`.

2. **Open a new terminal window that will be dedicated to the graphics server session.**

   On a Mac OS X client, you might need to start an X terminal window (`xterm`) with the Command-N key combination within the Mac OS X X11 application.

3. **In the same terminal window, open a Secure Shell session into the graphics server using `vglconnect`:**

```
client% /opt/VirtualGL/bin/vglconnect [vglconnect-option] user@graphics-server
VirtualGL Client v2.1 (Build 20071109)
Listening for SSL connections on port 4242
Listening for unencrypted connections on port 4243
Redirecting output to /home/susieq/.vgl/vglconnect-client-:0.0.log
```

Your *vglconnect-option* value depends on your security choice:

Security Choice	*vglconnect-option* Value
X11 Forwarding, VGL unencrypted	(no *vglconnect-option* value is used)
SSL-Encrypted VGL Images	(no *vglconnect-option* value is used)
X11 Forwarding, `ssh`-Encrypted VGL	**`-s`**

You can use `vglconnect -s` to create multilayered `ssh` tunnels. For instance, if the VirtualGL server is not directly accessible from the Internet, you can use `vglconnect -s` to connect to a gateway server, then use `vglconnect -s` again on the gateway server to connect to the VirtualGL server. Both the X11 and the VGL image traffic will be forwarded from the VirtualGL server through the gateway and to the client.

Replace *user* with your user account name on the graphics server. If your account name is the same on the current host as on the graphics server, then the *user***@** can be omitted. Replace *graphics-server* with the hostname (or IP address) of that graphics server.

4. **Within the `ssh` session, start any graphics application using `vglrun`:**

```
graphics-server% /opt/VirtualGL/bin/vglrun [vglrun-options] graphics-program [my-arguments]
[VGL] NOTICE: Automatically setting VGL_CLIENT environment variable to
[VGL]    100.200.30.45, the IP address of your SSh client.
```

Your *vglrun-options* value depends in part on your security choice:

Security Choice	*vglrun-option* Value
X11 Forwarding, VGL unencrypted	(no *vglrun-option* value is used)
SSL-Encrypted VGL Images	**`+s`**
X11 Forwarding, ssh-Encrypted VGL	(no *vglrun-option* value is used)

Replace *graphics-program* with your graphics program's executable file name, script name, or pathname. Provide any options or arguments to the graphics program at the end of the command line.

No action is required on the client as long as the SSL port traffic is not blocked by the client's firewall. By default, the client automatically accepts SSL or unencrypted connections.

Using VirtualGL From a Windows Client

A PC running Windows can be a client with Exceed 2006 or newer. For applications that use stereographic or transparent overlays, Exceed 3D is required on the client. The client desktop must be configured to display true color (24-bit pixels).

Instructions for installing and configuring VirtualGL on Windows are in "Installation on a Windows Client" on page 26, or the *VirtualGL User's Guide*. See "Related Documentation" on page xvii. Ensure that Exceed has been configured.

This section describes using VirtualGL's VGL Image Transport from a Windows client. This mode is not recommended for use on low-bandwidth or high-latency networks. Those networks require the Exceed (or Exceed 3D) X server to be installed on the Windows client.

▼ To Use VirtualGL From a Windows Client

1. **Start Exceed if it isn't already started.**

   Hover the mouse pointer over the Exceed taskbar icon and note the Exceed display number (for example, `Exceed 0.0 Multiwindow Mode`.)

2. **Open a new command prompt window and set the `DISPLAY` environment variable.**

```
C> set DISPLAY=:0.0
```

Replace *:0.0* with the Exceed display number.

If you only ever plan to use one Exceed session at a time, then you can set the `DISPLAY` environment variable in your global user environment (found in Control Panel->System->Advanced->Environment Variables).

3. **In that same command window, open a Secure Shell session into the graphics server using `vglconnect`:**

```
C> cd /d "c:\program files\virtualgl-version-build"
C> vglconnect [vglconnect-option] user@graphics-server
```

Your *vglconnect-option* value depends on your security choice:

Security Choice	*vglconnect-option* Value
X11 Forwarding; VGL unencrypted	(no *vglconnect-option* value is used)
SSL-Encrypted VGL Images	(no *vglconnect-option* value is used)
X11 Forwarding; `ssh`-Encrypted VGL	**`-s`**

`vglconnect -s` can be used to create multilayered `ssh` tunnels. For instance, if the VirtualGL server is not directly accessible from the Internet, you can use `vglconnect -s` to connect to a gateway server, then use `vglconnect -s` again on the gateway server to connect to the VirtualGL server. Both the X11 and the VGL image traffic will be forwarded from the VirtualGL server through the gateway and to the client.

Replace *user* with your user account name on the graphics server. Replace *graphics-server* with the hostname (or IP address) of that graphics server.

4. **Within the `ssh` session, start a graphics application using `vglrun`:**

```
graphics_server% /opt/VirtualGL/bin/vglrun [vglrun-options] graphics-program [my-arguments]
[VGL] NOTICE: Automatically setting VGL_CLIENT environment variable to
[VGL]    100.200.30.44, the IP address of your SSh client.
```

Your *vglrun-options* value depends in part on your security choice:

Security Choice	*vglrun-option* Value
X11 Forwarding, VGL unencrypted	(no *vglrun-option* value is used)
SSL-Encrypted VGL Images	**+s**
X11 Forwarding, `ssh`-Encrypted VGL	(no *vglrun-option* value is used)

Replace *graphics-program* with your graphics program's executable file, script name, or pathname. Provide any options or arguments to the graphics program at the end of the command line.

No action is required on the client as long as the SSL port traffic is not blocked by the client's firewall. By default, the client automatically accepts SSL or unencrypted connections.

Normal VirtualGL Messages

VirtualGL Client-Side Messages

When using the VGL Image Transport, `vglconnect` starts a `vglclient` daemon that awaits connections from VirtualGL applications (started with `vglrun`) from remote graphics servers. The VirtualGL client uses negligible resources until a server process connects to the client.

`vglconnect` prints messages identifying the ports used:

```
client% /opt/VirtualGL/bin/vglconnect graphics-server
VirtualGL Client v2.1 (Build 20071109)
Listening for SSL connections on port 4242
Listening for unencrypted connections on port 4243
Redirecting output to /home/susieq/.vgl/vglconnect-client-:0.0.log
```

If `vglconnect` is used more than once from the same client (for example, to connect to a different graphics server), it might tell you that `vglclient` is already running on this X display:

```
client% /opt/VirtualGL/bin/vglconnect myserver
vglclient is already running on this X display and accepting SSL
   connections on port 4243.
vglclient is already running on this X display and accepting unencrypted
   connections on port 4242.
```

This output is fine. `vglclient` is designed to stay active in the background and available for subsequent connection from any number of remote applications.

VirtualGL Server Messages

`vglrun` marks its innocuous messages with `NOTICE`. If the `VGL_CLIENT` environment variable is not set, but `ssh` environment variables are set, VirtualGL will set `VGL_CLIENT` from the `ssh` variables, and print this message to remind you, in case you intended to do something else:

```
[VGL] NOTICE: Automatically setting VGL_CLIENT environment variable to
[VGL]    100.200.30.44, the IP address of your SSh client.
```

Troubleshooting VirtualGL

This section provides a starting point for analyzing problems with the use of VirtualGL's VGL Image Transport.

▼ To Verify X Server Access

You can verify the ability of an application on the graphics server to access your client's X server.

- **Start the `xclock` application.**

- For the Solaris Operating System, type:

```
my_server% /usr/openwin/bin/xclock &
```

- For Linux, type:

```
my_server% xclock &
```

The appearance of the `xclock` application verifies access.

Tip – If the clock does not appear, although the `$DISPLAY` value is correct, the client might be too secure to allow remote TCP access. The client host's system administrator can configure the client's X server to allow remote TCP access.

Could Not Connect

If a `vglclient` had been running on this client but is no longer running and active, you might see this server message:

```
[VGL] ERROR: Could not connect to VGL client.  Make sure that vglclient is
[VGL]    running and that either the DISPLAY or VGL_CLIENT environment
[VGL]    variable points to the machine on which vglclient is running.
```

This message indicates that the `vglrun` process could not communicate with the `vglclient` process on the client. Sometimes additional messages follow, such as:

```
[VGL] ERROR: in rrsocket.cpp--
[VGL]    226: Connection refused
```

▼ To Reconnect to Your `vglclient`

When your server's VirtualGL cannot connect to your `vglclient`, follow these steps on your client:

1. **Run `vglclient -kill` to make sure that there aren't any `vglclient` processes running.**

   This action might print the process ID of a `vglclient` process being terminated.

2. **Restart the `vglclient` by using `vglconnect` with the `-force` option.**

```
client% /opt/VirtualGL/bin/vglconnect -force user@my-server
```

If your application (or a specific sequence of actions) repeatedly causes `vglclient` to crash, file a bug report.

Manually Using TurboVNC

If TurboVNC or a web browser is used for the client, this section provides detailed information. This method is well suited for a slow or high-latency network (for example, the Internet) but also performs well on a faster network, such as a LAN. TurboVNC also enables multiple clients to share graphics applications, aiding in collaboration.

On a Solaris or Linux host, you can read the TurboVNC man pages with a command such as:

```
man -M /opt/TurboVNC/man vncserver
```

You can read additional related man pages by substituting the following for `vncserver`:

- `Xvnc`
- `vncviewer`
- `vncconnect`
- `vncpasswd`

On a Mac OS X host, the only man page is for `vncviewer`.

Note – For Windows, after installing TurboVNC, use the embedded help feature. The icon for help is a question mark in the upper-right corner of the TurboVNC viewer window.

There are two components to be started:

- TurboVNC (Virtual Network Computing) server running on the graphics server
- One or more clients viewing the TurboVNC session using one of these types of TurboVNC client viewing software:
  - The Java based Web VNC viewer software runs within a web browser on the client (simple – requires no installation)

- A dedicated TurboVNC `vncviewer` client software component (much better-performing)

Some users can use the TurboVNC `vncviewer` client while others sharing the same VNC session use the Java based Web VNC viewer.

TurboVNC Process Overview

To run your application within a TurboVNC session hosted on the graphics server, you use these six steps:

1. Set TurboVNC's password using `vncpasswd`.

Note – Do this step upon the initial execution. The step does not need to be repeated for later runnings.

---

2. Access the graphics server.
3. Start a TurboVNC session on the graphics server.
4. Start a TurboVNC viewer on your client (host). Additional viewers can be started by collaborators. This step is dependent on which TurboVNC viewer is used.
5. Start your application within the TurboVNC session on the graphics server. Invoke the application using `vglrun`, so the application is under the control of VirtualGL.
6. Eventually, terminate the TurboVNC server session (and all clients).

These steps differ slightly depending on whether you choose to use the `vncserver` command or the `RUN.vncserver` script, which is part of the optional Sun Grid Engine Additions. `RUN.vncserver` might be easiest, but requires the same home directory to be shared by the client host and the graphics server.

The procedures to follow for either choice are listed in TABLE 3-1. The third and fourth procedures in each sequence are different.

TABLE 3-1 Procedure Sequence for Manually Using TurboVNC

Procedure Order	With the `vncserver` Command	With the `RUN.vncserver` Script
1	"To Select a TurboVNC Password" on page 44	"To Select a TurboVNC Password" on page 44
2	"To Access the Graphics Server" on page 45	"To Access the Graphics Server" on page 45
3	"To Start the TurboVNC Server Session" on page 46	"To Start the TurboVNC Server Session Using `RUN.vncserver`" on page 52

TABLE 3-1 Procedure Sequence for Manually Using TurboVNC

Procedure Order	With the `vncserver` Command	With the `RUN.vncserver` Script
4	"To Start a TurboVNC Viewer and Connect to Your TurboVNC Session" on page 46	"To Connect a Viewer to Your `RUN.vncserver` Session" on page 52
5	"To Start a Graphics Application Within a TurboVNC Session" on page 49	"To Start a Graphics Application Within a TurboVNC Session" on page 49
6	"To Terminate the TurboVNC Session" on page 50	"To Terminate the TurboVNC Session" on page 50

Manually Using the `vncserver` Command

The following sections detail the six steps for manually using the Sun Shared Visualization 1.1 software with the `vncserver` command.

▼ To Select a TurboVNC Password

Before running the TurboVNC server for the first time, you should select a TurboVNC password, which may differ from your login password.

- **Start** `vncpasswd`**:**

```
client% /opt/TurboVNC/bin/vncpasswd
Using password file /home/susieq/.vnc/passwd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
```

If `/opt/TurboVNC/bin` is in your `$PATH`, then you can start `vncpasswd`. The view-only password is an alternate password to be given to a collaborator you want to enable to join your TurboVNC session. This collaborator can only view your session, not move the mouse, nor enter keyboard or mouse events. The TurboVNC password (and any view-only password) will be used by all sessions started by this user using the same `$HOME` directory. The password can be changed before any session.

▼ To Access the Graphics Server

- **Take one of the following actions:**
  - On a Solaris, Linux, or Mac OS X client:

    Open a new terminal window. Within the window, use `ssh` to access the graphics server:

```
client% ssh user@my-server
Password:
Last login: Wed May 12 13:33:52 2006 from client
Sun Microsystems Inc. SunOS 5.10      Generic January 2005
```

  - On a Windows client

    Open a command prompt window. Within the window, use `putty` to access the graphics server:

```
C> "c:\program files\turbovnc\putty" user@graphics-server
Password:
Last login: Wed May 12 13:33:52 2006 from client
Sun Microsystems Inc. SunOS 5.10      Generic January 2005
```

Replace *user* with your user account name on the graphics server. Replace *graphics-server* with the hostname (or IP address) of that graphics server.

The `DISPLAY` environment variable on the graphics server shell is irrelevant to the TurboVNC session you are about to establish. The TurboVNC server session is itself an X server. Within the TurboVNC session all windows have a `DISPLAY` value starting with *my-server*.

▼ To Start the TurboVNC Server Session

- **Start the TurboVNC server session on the graphics server host using the** `vncserver` **script.**

  If the graphics server is running Solaris software, the initial invocation looks like the following with Shared Visualization release 1.1.1:

```
my_server% /opt/TurboVNC/bin/vncserver

New 'X' desktop is my_server:1

Creating default startup script /home/susieq/.vnc/xstartup.turbovnc
Starting applications specified in /home/susieq/.vnc/xstartup.turbovnc
Log file is /home/susieq/.vnc/server:1.log
```

  If you add `/opt/TurboVNC/bin` to your `$PATH`, you can start `vncviewer` without typing the path.

  You can specify a size (in pixels) for the `vncserver`'s created desktop using its `-geometry w x h` option. If the size is too small, applications might not fit. But if the size is too large, you are not able to display all of the desktop at once on your client, so you will get scroll bars.

  The terminal window shell displays the `vncserver`'s output. The key line of output displays the TurboVNC display name (and is set off by blank lines). Note the server name (`my_server` in this example) and X display number (1 here). You can think of the values, separated by the colon, as the display name, such as `my_server:1` in this example.

  The `ssh` session to the graphics server can now be exited, if desired. However, you might want to use this session as a reminder to eventually kill the VNC server session.

▼ To Start a TurboVNC Viewer and Connect to Your TurboVNC Session

This procedure differs, based on which TurboVNC viewer you use on your client:

- The Java based TurboVNC viewer software runs within a web browser on the client (simple)
- A dedicated `vncviewer` client software component (much better-performing)

Once your TurboVNC viewer is connected to your TurboVNC session, within this TurboVNC X session you can create multiple terminals (shell windows) and start graphics applications.

1. **Decide which VNC viewer you will use.**
   - To use a simple Java-based TurboVNC client viewer in your web browser, continue with Step 2.
   - To use a TurboVNC viewer that performs better, continue with Step 3.
2. **Connect your web browser to your TurboVNC session.**

   In your web browser, type the URL containing the server name and the port number, which is 5800 + the display number noted previously.

   For the previous example (`my_server:1`, where the display number is `1`), the URL is `http://my_server:5801`. The web server displays a separate TurboVNC Java applet window. This window enables you to set options. (You also can change options using the Options button at the top of the TurboVNC session window.) This window will prompt you for the TurboVNC password before it enables you to view the TurboVNC session.

   After the viewer is enabled, continue with the next procedure, "To Start a Graphics Application Within a TurboVNC Session" on page 49.
3. **Locate your TurboVNC server session's display name.**

   This display name is the graphics server name and the X display number, separated by a colon. (The name `my_server:1` is used in the examples in this procedure.)

   This name is available when you start your TurboVNC session.

   You will include this display name as an option to the TurboVNC viewer, so the viewer can connect to your TurboVNC session.
4. **Start a TurboVNC viewer connected to your TurboVNC session.**

   VNC offers a client program specifically for use on the client host as a remote TurboVNC viewer, which can offer better window system integration and much better performance than the web browser technique.

   On a Solaris, Linux, or Mac OS X client, the TurboVNC viewer (`vncviewer`) is in `/opt/TurboVNC/bin/`. If you add this directory to your `$PATH` variable, you can start `vncviewer` without typing the path.

- On a Solaris, Linux, or Mac OS X client, your command and the `vncviewer` output might be the following:

```
client% /opt/TurboVNC/bin/vncviewer my_server:1
Connected to RFB server, using protocol version 3.7
Enabling TightVNC protocol extensions
Performing standard VNC authentication
Password:
VNC authentication succeeded
```

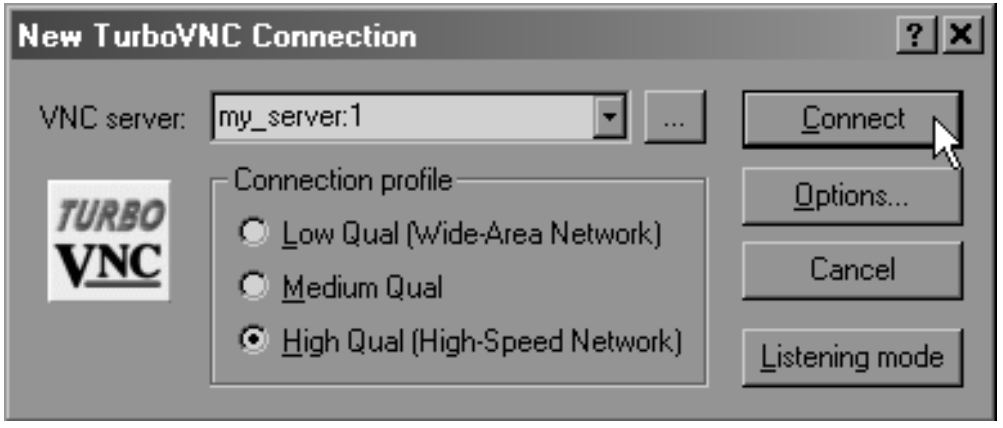
  If you do not put your graphics server's VNC display name on the `vncviewer` command line, a small `vncviewer` window will prompt for it. Enter the display name and press the Return key.

  The TurboVNC viewer prompts you for your TurboVNC password and then enables you to view the TurboVNC session.

- On a Windows client, select `TurboVNC Viewer` in the TurboVNC Start Menu group.

a. Select a connection profile in the dialog.

FIGURE 3-1 TurboVNC Connection Dialog on a Windows Client



b. When prompted, enter your password and click OK.

For more information on connection profiles, which enable you to establish settings that balance quality and performance, see "TurboVNC Image Encoding Protocols and Dynamic Quality/Performance Tradeoff" on page 116.

A TurboVNC session window appears on your client host. This client window views the TurboVNC X server on the TurboVNC server host. Within this TurboVNC session, you can launch X Windows applications that will run on the server host.

▼ To Start a Graphics Application Within a TurboVNC Session

Within the TurboVNC session, you might type commands to the graphics server's shell windows normally. However, when you are ready to run a graphics application, you must use VirtualGL's `vglrun` command. `vglrun` interposes between the application and the GLX library so `vglrun` can read completed images from the graphics accelerator and pass the images to the TurboVNC server session. The `vglrun` command can be in your `$PATH`. Otherwise, you need to use a full path to the `vglrun` command.

VirtualGL avoids compressing the graphics images VirtualGL gives to the TurboVNC session on the same host. TurboVNC compresses images it sends to its viewer. TurboVNC also sends images to clients only as fast as the client can display the images. Therefore, the VNC session does not necessarily send every updated frame to every client.

- **Use a `vglrun` command to start your graphics application.**

  For example, enter this command from within a terminal window in the TurboVNC session:

```
my_server% /opt/VirtualGL/bin/vglrun myprogram
```

Note – If you have used the `RUN.vncserver` script, the `vglrun` command should be in your `$PATH`, since `RUN.vncserver` added `vglrun`'s directory. But the `.cshrc` or `.profile` in your `$HOME` might have overridden the `$PATH` the file inherits. In that case, you need to use a full path to the `vglrun` command.

If you attempt to run an OpenGL application from within your TurboVNC session without remembering to use `vglrun` (but with `$DISPLAY` directing the application to your TurboVNC session), you might get an error message such as:

```
Xlib:  extension "GLX" missing on display "my_server:1.1"
```

▼ To Terminate the TurboVNC Session

Do not forget to save your work and terminate the TurboVNC session when you are done with it.

Exiting the viewer by quitting your web browser, leaving the TurboVNC page, or exiting the `vglviewer` does not terminate the TurboVNC server session. When you have saved your work, you must cause the TurboVNC session and all TurboVNC foreground processes to exit.

- **Take one of the following actions:**
  - Use the TurboVNC session's window manager logout procedure.
  - From within the TurboVNC session or from any session to the same graphics server, issue a `kill` command to the graphics server host. Include the display number noted upon startup of the TurboVNC session (`1` in the example):

```
my_server% /opt/TurboVNC/bin/vncserver -kill :1
```

When the TurboVNC server session exits, `vglviewer` exits, and a web browser viewer prompts for a session password.

To list the X display numbers and process IDs of all TurboVNC server sessions that are currently running under your user account on this system, type:

```
my_server% /opt/TurboVNC/bin/vncserver -list
```

Manually Using the `RUN.vncserver` Script

Even without submitting a job to Sun Grid Engine, the `RUN.vncserver` script that is an optional part of Shared Visualization 1.1 server installation can help start the TurboVNC server session. This script is available to clients that have mounted the Sun Grid Engine installation from the grid's NFS server.

This process is nearly identical to the process described in "Manually Using the `vncserver` Command" on page 44. However, a file holds the TurboVNC session's `DISPLAY` value, relieving the user of this burden. Because the file is under the user's `$HOME`, it also has these two disadvantages:

- The client host and the graphics server are assumed to share the same home directory.
- The user can have no more than one script-started TurboVNC session active at a time.

The first two procedures in the sequence are the same whether you are using the `vncserver` command or the `RUN.vncserver` script:

- First, to select a TurboVNC password, see "To Select a TurboVNC Password" on page 44. This procedure must be performed sometime prior to using the `RUN.vncserver` command.
- Second, to access the graphics server, see "To Access the Graphics Server" on page 45.

When you use the `RUN.vncserver` script, the third and fourth procedures in the sequence are different and easier:

- Third, to start the TurboVNC session, see "To Start the TurboVNC Server Session Using `RUN.vncserver`" on page 52.
- Fourth, to connect a viewer to the session, see "To Connect a Viewer to Your `RUN.vncserver` Session" on page 52.

The final two procedures are the same in both cases:

- Fifth, to start an application, see "To Start a Graphics Application Within a TurboVNC Session" on page 49.
- Finally, to terminate the TurboVNC or `RUN.vncserver` session, see "To Terminate the TurboVNC Session" on page 50.

The procedures included in this section are the two that are different when you use `RUN.vncserver`.

▼ To Start the TurboVNC Server Session Using `RUN.vncserver`

Note – This section substitutes for "To Start the TurboVNC Server Session" on page 46.

---

1. **Type the `RUN.vncserver` command:**

```
my_server% /gridware/sge/graphics/RUN.vncserver &
```

The output and any errors from the `RUN.vncserver` script is in `$HOME/vncserver.log`. If your personal configuration files, such as `$HOME/.profile` or `$HOME/.cshrc` do not override the `$PATH` or `csh $path` established by the `RUN.vncserver` script, then `vglrun` (used to start a graphics application) is in your `$PATH`.

2. **Add the configuration files to the `$PATH` the files receive, rather than replacing that path.**

Your grid can have a different script for this purpose, specific to your environment.

Note – The files written by the `RUN.vncserver` script are in your `$HOME` on the execution host (graphics server), if that differs from your `$HOME` on your client (host).

After starting the VNC server session, the `ssh` session to the graphics server can now be exited, if desired. However, you might want to use this session as a reminder to eventually exit the VNC session.

▼ To Connect a Viewer to Your `RUN.vncserver` Session

Note – This section substitutes for "To Start a TurboVNC Viewer and Connect to Your TurboVNC Session" on page 46.

This step depends on your TurboVNC viewer. Either viewer should be convenient, as long as the graphics server host and the client host share the same `$HOME` directory.

- **Connect your web browser to your TurboVNC session.**

  The `RUN.vncserver` script creates files in your `$HOME` directory starting with `vnc_`. The file `$HOME/vnc_url` should redirect your browser to the execution server and port number for your TurboVNC session. If your web browser expands `$HOME`, you could simply enter (or select a bookmark for) `$HOME/vnc_url` or `file://$HOME/vnc_url`. If neither of these methods work, you can expand `$HOME` yourself and type `file://` and your home directory followed by `/vnc_url` (for example, `file:///home/susieq/vnc_url`). This action redirects your browser to the URL contained in your `vnc_url` file.

  Note – The files written by the `RUN.vncserver` script are in your `$HOME` on the execution host (graphics server), if that `$HOME` differs from your `$HOME` on your client (host).

  You can also view your `$HOME/vnc_url` file and use your browser to view the URL contained in that file (for example, `http://my_server:5802`).

  The web page prompts you for the TurboVNC password and then enables you to view the TurboVNC session. Within this TurboVNC X session, you can create multiple terminals (shell windows) and start graphics applications. See "To Start a Graphics Application Within a TurboVNC Session" on page 49.

- **Start a TurboVNC viewer connected to your `RUN.vncserver` session.**

  The script saves the graphics server name and port number in the file `$HOME/vnc_server` in a format useful to the TurboVNC viewer. You can start the TurboVNC viewer on your client (host) by appending `` `cat $HOME/vnc_server` `` as an option to your `vncviewer` starting. If your client is running Linux, your command might be:

```
client% /opt/TurboVNC/bin/vncviewer `cat $HOME/vnc_server`
```

  You can make a shell `alias` for this command.

Security With TurboVNC

Normally, the connection between the TurboVNC server session and the TurboVNC viewer is completely unencrypted, but securing that connection can be easily accomplished by using the port forwarding feature of Secure Shell (`ssh`).

▼ To Secure the Connection Between the TurboVNC Server Session and Viewer

1. **Start a TurboVNC session on the server.**
2. **Open a new `ssh` connection into the server with one of the following command lines:**
   - On a Solaris, Linux, or Mac OS X client:

```
client% ssh -L 5900+n:localhost:5900+n  user@graphics-server
```

   - On a Windows client:

```
C> "c:\program files\turbovnc\putty" -L 5900+n:localhost:5900+n  user@graphics-server
```

   In either case, replace *user* with your user account name on the graphics server. Replace *graphics-server* with the hostname (or IP address) of that graphics server.

   Replace *5900+n* with the sum of 5900 and the X display number of the TurboVNC server session to which you want to connect.

   For instance, if you want to connect to display :1 on server `my_server` using user account `my_user`, you type:

```
client% ssh -L 5901:localhost:5901 my_user@my_server
```

3. **Start the TurboVNC viewer and point it to `localhost:`*n* (`localhost:1` in the preceding example).**

Performance Notes on TurboVNC and `ssh`

For LAN connections and other high-speed networks, tunneling the TurboVNC connection over `ssh` will reduce performance by as much as 20 to 40 percent. But for connections such as wide-area networks and broadband, there is little or no performance penalty for using `ssh` tunneling with TurboVNC.

Performance and Measurement

This section describes controlling and measuring VirtualGL performance. Additional information is in Appendix A and Appendix B.

Spoiling

By default, VirtualGL discards frames when it is already busy sending a frame. This behavior, called *spoiling*, allows the most recent frame to appear sooner at the client. The most recent frame need not wait behind older frames. However, this behavior means that the application and graphics accelerator spend time producing frames that will be discarded. The `-spoil` option to `vglrun` disables spoiling. This behavior forces every distinct frame that is produced by the application to be transported to the client and displayed. Therefore, this behavior slows the application to the speed at which the frames can be compressed, transmitted, decompressed, and displayed. In this way, the application's stated performance matches the client's results.

The `vglrun +profile` option outputs (typically, to the `vglconnect` log file) lines that show its performance for compression, transmission, decompression, display, and overall (total) performance in frames per second (fps) and megapixels per second.

For example, an application with spoiling enabled might produce 60 frames per second, of which 10 are displayed for the user. The application reports that its performance is 60 frames per second. However, the `vglrun` profile output shows a total of only 10 fps.

If you rerun `vglrun` with `-spoil`, the application's performance will match the profile performance of `vglrun`. Because resources are not spent producing discarded frames, more than 10 fps might now reach the client (depending on where the bottleneck was). This example might now produce, transmit, and display to the client 20 fps.

TurboVNC also spoils frames, but its behavior cannot be controlled by VirtualGL options or environment variables. Each VNC client pulls a frame from the server when it is ready, rather than the server pushing images to the clients when the image has been updated by an application.

TurboVNC Quality Controls

TurboVNC supports both static and dynamic controls over quality and performance tradeoffs. See "TurboVNC Image Encoding Protocols and Dynamic Quality/Performance Tradeoff" on page 116.

CHAPTER 4

Using Sun Grid Engine to Start the Sun Shared Visualization 1.1 Software

Topics in this chapter include:

Sun Grid Engine Overview

Sun Grid Engine performs resource management. Sun Shared Visualization 1.1 software extends resource management for graphics servers to allocate graphics resources, as well as CPUs, memory, and other components. In an environment that has multiple execution servers or multiple graphics accelerators on a host, Sun Grid Engine (SGE) can select a suitable, lightly-loaded server to run your application. Sun Grid Engine can also select a lightly-loaded graphics device on that server. Sun Grid Engine software starts applications on that execution server, so you need not log in to the server.

Job scripts can specify options to Sun Grid Engine, which simplifies the task for its users. In an environment with heterogeneous execution servers, these options could specify which processor types and operating systems are capable of running the application. Additionally, the scripts can be customized for your Sun Grid Engine environment.

Preparing to Use Sun Grid Engine With VirtualGL

This section describes using the VirtualGL VGL or Sun Ray Image Transport from a Solaris, Linux, or Mac OS X client. Do not use this mode on low-bandwidth or high-latency networks.

You must first start the client's X server and log in to the client before performing procedures in this chapter.

On a Windows client, you also must install and configure Exceed. Instructions for installing and configuring VirtualGL on Windows are in "Installation on a Windows Client" on page 26, or the *VirtualGL User's Guide*. (To access that document, see "Related Documentation" on page xvii.) Ensure that Exceed has been configured.

Determining if Your Client's X Server Allows Remote TCP Connections

These procedures require that your client's X server allow remote TCP connections (that is, into your X server from the application execution server chosen by Sun Grid Engine).

To enhance security, some newer Linux and Solaris distributions (in particular, at least Solaris 10 release 11/06) do not by default allow TCP connections into the X server. Such systems cannot be used as clients with these procedures, unless the systems are reconfigured to allow X11 TCP connections. Discuss this situation with your system administrator.

Note – If the system administrator decides to reconfigure the system, the following must be done. On a Solaris client, as root, remove the `-nolisten tcp` option from the Xserver invocation line, which is at the end of `/etc/dt/config/Xservers`. If that file doesn't exist, create it, as root, by copying `/usr/dt/config/Xservers` to `/etc/dt/config/Xservers` and give the `/etc` copy write permission. See the `Xserver`(1) manpage, which is under `/usr/openwin/man` in a Solaris installation.

If allowing remote X connections is not feasible, consider using SGE with TurboVNC instead. See "Submitting Sun Grid Engine TurboVNC Jobs" on page 67 for instructions on those processes.

Determining if Your Client Host Can Be a Sun Grid Engine Submit Host

The procedure for using Sun Shared Visualization 1.1 software with Sun Grid Engine depends on whether your client is a Sun Grid Engine submit host. If so, you will submit jobs directly from your client. If not, you will first connect to a submit host and submit your job from there.

Sun Grid Engine Submit Host Clients

If your client is permitted to be an SGE submit host, it will need to NFS mount the SGE installation, as described in Appendix B. This mount must enable your client to use the same `$SGE_ROOT` as is used on the rest of the grid. (The default `SGE_ROOT` is `/gridware/sge`.)

Windows Submit Hosts

Windows submit hosts are more difficult to configure successfully for Sun Grid Engine than Solaris or Linux submit hosts. To configure a Windows host for Sun Grid Engine currently requires:

- Obtaining and installing Microsoft's Services for UNIX (SFU) software, which gives the Windows host a POSIX (UNIX-like) environment.
- Configuring user name mapping.
- NFS mounting from the grid's queue master. This process is more difficult on Windows, especially to enable `qrsh`, which requires `set-uid` permission.

These tasks require an experienced Sun Grid Engine administrator. See the current *Sun N1 Grid Engine 6.1 Installation Guide*, part number 820-0697, including its appendices.

Clients That Are Not Sun Grid Engine Submit Hosts

If your client is not an SGE submit host, you must first connect to a submit host. This host need not be a graphics server but must be in the same grid with the graphics servers. From there, you submit your Sun Grid Engine job. Sun Grid Engine assigns your job to a lightly-loaded execution host (in keeping with your job's stated requirements). Then, if the job requested graphics, Sun Grid Engine assigns a graphics accelerator (device or X display) to your job. When your job runs, its X applications connect back to your client. The submit host and the graphics execution hosts must share the same home directories and reside in the same domain.

This procedure requires that your client's X server allow remote TCP connections, because you don't know which execution host SGE will select for your job. Both the application's X11 traffic and the VirtualGL image stream are unencrypted.

▼ To Prepare to Use VirtualGL From a Windows Client

This section describes using VirtualGL's VGL Image Transport from a Windows client. Do not use this mode on low-bandwidth or high-latency networks. This mode requires the Exceed (or Exceed 3D) X server to be installed and configured on the Windows client.

1. **Start Exceed if it isn't already started.**

   Hover the mouse pointer over the Exceed taskbar icon and make a note of the Exceed display number (for example, `Exceed 0.0 Multiwindow Mode`).

2. **Open a new command prompt window and set the `DISPLAY` environment variable.**

   ```
   C> set DISPLAY=:0.0
   ```

   Replace *:0.0* with the Exceed display number.

   If you only ever plan to use one Exceed session at a time, then you can set the `DISPLAY` environment variable in your global user environment (found at Control Panel->System->Advanced->Environment Variables).

3. **Use the same command window to open a Secure Shell session into the graphics server using `vglconnect`, described later.**

Submitting Sun Grid Engine Graphics Jobs

This section contains two procedures:

- "To Submit Sun Grid Engine Graphics Jobs if Your Client Is Also a Sun Grid Engine Submit Host" on page 61
- "To Submit Sun Grid Engine Graphics Jobs if Your Client Is Not a Sun Grid Engine Submit Host" on page 62

▼ To Submit Sun Grid Engine Graphics Jobs if Your Client Is Also a Sun Grid Engine Submit Host

If your client is also a Sun Grid Engine submit host in the same grid with one or more graphics servers, you can submit your SGE job directly from your client. However, before doing so, you will use `vglconnect -k` to give the graphics execution hosts access to your X display.

Sun Grid Engine will assign your job to a lightly-loaded execution host (in keeping with your job's stated requirements), and then assign a graphics accelerator (device or X display) to your job. When your job runs, its X applications will connect back to your client. The submit host and the graphics execution hosts must share the same home directories and reside in the same domain.

Your client's X server must allow remote TCP connections, because you don't know which execution host SGE will select for your job. Both the application's X11 traffic and the VirtualGL image stream are unencrypted.

1. **Within any terminal window on your client, use `vglconnect -k` to enable a remote X program to access your client's X server.**
   - On a Solaris, Linux, or Mac OS X client (or on a Sun Ray client, with a Solaris or Linux Sun Ray server):

```
client% /opt/VirtualGL/bin/vglconnect -k
```

   - On a Windows client:

```
C> cd /d "c:\program files\virtualgl-version-build"
C> vglconnect -k
```

   Note that no user or server is named on the `vglconnect -k` command line.

2. **Within any terminal window on your client, set up the Sun Grid Engine environment.**
   - `tcsh/csh` users set up environment variables using:

```
client% source /gridware/sge/default/common/settings.csh
```

   Substitute `/gridware/sge` with your value for `$SGE_ROOT`.

- sh/bash/ksh users use the . command:

```
$ . /gridware/sge/default/common/settings.sh
```

  Substitute /gridware/sge with your value for $SGE_ROOT.

The qstat -f command shows you available Sun Grid Engine execution hosts, queues, and any active Sun Grid Engine jobs. See Appendix C for more information on Sun Grid Engine commands.

3. **Within the same terminal window (with the SGE environment), submit SGE jobs using qsub or qrsh.**

   See "Using Sun Grid Engine to Start Your Graphics Application" on page 63.

▼ To Submit Sun Grid Engine Graphics Jobs if Your Client Is Not a Sun Grid Engine Submit Host

1. **Identify the submit host to which you will connect.**
2. **Open a new terminal window that will be dedicated to the session on this submit host.**
3. **In the same terminal window, use vglconnect -x to start a session to this submit host.**
   - On a Solaris, Linux, or Mac OS X client:

```
client% /opt/VirtualGL/bin/vglconnect -x user@submit-host
```

   - On a Windows client:

```
C> cd /d "c:\program files\virtualgl-version-build"
C> vglconnect -x user@submit-host
```

   Replace *user* with your user account name on the graphics server. If your account name is the same on the current host as on the graphics server, then you can omit the *user@* portion. Replace *submit-host* with the hostname (or IP address) of that submit host.

4. **From within the ssh session, set up the Sun Grid Engine environment.**

- `tcsh/csh` users set up environment variables using:

```
submit_host% source /gridware/sge/default/common/settings.csh
```

  Substitute `/gridware/sge` with your value for `$SGE_ROOT`.

- `sh/bash/ksh` users should use the `.` command:

```
$ . /gridware/sge/default/common/settings.sh
```

  Substitute `/gridware/sge` with your value for `$SGE_ROOT`.

The `qstat -f` command shows you available Sun Grid Engine execution hosts, queues, and any active Sun Grid Engine jobs. See Appendix C for more information on Sun Grid Engine commands.

5. **From within the `ssh` session, submit SGE jobs using `qsub` or `qrsh`.**

   See the next section, "Using Sun Grid Engine to Start Your Graphics Application" on page 63.

Using Sun Grid Engine to Start Your Graphics Application

There are two ways to use Sun Grid Engine to start your graphics application:

- Using an application script.

  Your system administrator might have prepared a script for you that submits to Sun Grid Engine a job that executes your desired application. In this case, you can invoke such a script.

- Starting a graphics application using a Sun Grid Engine job script.

  There are Sun Grid Engine job scripts you can submit to Sun Grid Engine that set Sun Grid Engine options and when your job starts executing, starts your application. Submitting such a Sun Grid Engine job script could be as simple as:

  - Using `qrsh -b no`:

```
submit_host% qrsh -b no /path/to/my-application-script
```

- Or, using `qsub -now y`:

```
submit_host% qsub -now y /path/to/my-application-script
```

See "Differences in qsub and qrsh Command Options" on page 128 for differences between these two commands, and an explanation of why each needs options to be appropriate for graphics job scripts.

Note – To run Sun Grid Engine commands without the full path, ensure that the `$SGE_ROOT` is in your `PATH` variable value.

You can override or add additional Sun Grid Engine options, such as:

```
submit_host% qsub -now y -l h_rt=3:0:0 /path/to/my-application-script
```

- Ask your system administrator if the job script starts `vglrun` implicitly when the script detects starting in a Sun Grid Engine graphics server environment. (An example of a job script that does detection is in "Example Sun Grid Engine Job Script" on page 129. Sun Grid Engine starts the job with the `VGL_DISPLAY` environment variable set. The job script uses `vglrun` to start the application.)
- If the job script does not start `vglrun` implicitly, an application that requires graphics resources turns this order around. The Sun Grid Engine job starts `vglrun`, and `vglrun` starts the application script within a VirtualGL environment.

  In general, the syntax of this startup is one of:

```
qsub -now y [SGE-options] /opt/VirtualGL/bin/vglrun [vglrun-options] application-name [app-options]
qrsh -b n   [SGE-options] /opt/VirtualGL/bin/vglrun [vglrun-options] application-name [app-options]
```

  You might also want or need additional *SGE-options* (that is, `qsub` or `qrsh` options), such as an option that specifies the architecture of the graphics server that Sun Grid Engine selects as your execution host.

  Though `vglrun` does not know what execution host architecture is appropriate for your application, `vglrun` does know that Sun Grid Engine `vglrun` jobs need graphics and need to save your `DISPLAY` or `SSH_CLIENT` environment variable values. `VGL_*` environment variable values are also saved.

  `vglrun` specifies these options for you, which simplifies the startup (as long as you are submitting with `-b n` (binary no) option, which is the default for `qsub` but not for `qrsh`).

The following example startup indicates that the application can run on any Linux host:

```
submit_host% qsub -now y -arch "lx24-*" /opt/VirtualGL/bin/vglrun /path/to/my_application
```

In this case, note that Sun Grid Engine scans the `vglrun` script for Sun Grid Engine options but does not scan the application script that is an argument to `vglrun`.

Note – The architectures named `lx24-x86` and `lx24-amd64` are used on Linux 2.4 kernels and also on Linux 2.6 kernels.

Sun Grid Engine requires a full path to the job's application or script. Sun Grid Engine does not search your `PATH`. Also, the path to a job's script must be valid on the submit host, because Sun Grid Engine reads the script at submission time. (SGE saves a copy of the job script with the job, and executes its copy at job execution time.)

`vglrun` does use `$PATH` to find its argument. However, the default `PATH` for a Sun Grid Engine job is very limited.

Easing Graphics Job Submission Using `alias`

If you do not provide job-specific `qsub` options, you can make a shell `alias` for `qsub` and `vglrun`, such as (`csh` or `tcsh` syntax):

```
submit_host% alias qrsh_vgl 'qrsh -b no  /opt/VirtualGL/bin/vglrun'
submit_host% alias qsub_vgl 'qsub -now y /opt/VirtualGL/bin/vglrun'
```

These aliases must be invoked with your application added at the end, such as:

```
submit_host% qrsh_vgl /path/to/my-application  my-option
```

If a graphics job script does not start `vglrun` implicitly and you do not start `vglrun` manually, the graphics application will attempt to use the GLX (OpenGL for X) remote graphics technique described in "Remote X Server Graphics" on page 3. This attempt will be successful if `$DISPLAY` directs the application to your client's X server and if your client's X server supports the GLX graphics extension. Even in this case, GLX can be far slower than VirtualGL, especially for large graphics models.

Graphics Job Submission Without a Job Script

If it is necessary to submit a job without any job script (that is, neither using an application-specific job script nor the `vglrun` generic job script), then you must specify all the required *SGE-options* on the command line. The options probably include:

SGE Option	Meaning and Purpose
`-l gfx=1`	Need 1 graphics resource. May be comma-separated list of resources.
`-l arch=`*value*	Specify required architecture (operating system and processor) value
`-N` *JobName*	Job is named *JobName.* (Job output files start with the *JobName*).
`-v DISPLAY`	Save current `DISPLAY` environment variable value with the job. Environment variable names may be a comma-separated list.
`-v SSH_CLIENT`	Save current `SSH_CLIENT` environment variable value with the job. (When `SSH_CLIENT` is set but `VGL_CLIENT` is not set, VirtualGL will determine the actual client from the `SSH_CLIENT` value.)

You might also wish to specify other options, such as:

SGE Option	Meaning and Purpose
`-j y`	"Join Yes" (`qsub` only): Joins standard error into output file.
`-v VGL_CLIENT`	Saves current `VGL_CLIENT` environment variable value with the job.
`-v VGL_GAMMA`	Saves current `VGL_GAMMA` environment variable value with the job.
`-q` *queueName*	Requires SGE queue *queueName.*

If you regularly use other `VGL_` environment variables, you might want their values also saved with the job.

The following is an example of a raw job submission:

```
submit_host% qsub -now y -l gfx=1,arch=lx24-amd64 -N fun -v DISPLAY,SSH_CLIENT \
fun_application
```

This job submission is sufficient to allocate a graphics resource, but `vglrun` must still be run for any graphics rendering to be directed to that graphics resource. You can run `vgerun` later if `fun_application` attempts to invoke `true_application`. The user could create a script named `true_application` that actually invokes the real `true_application` under control of VirtualGL:

```
/opt/VirtualGL/bin/vglrun /path/to/true-application
```

Submitting Sun Grid Engine TurboVNC Jobs

Using Sun Grid Engine simplifies the process of running your application within a TurboVNC session that is hosted on the graphics server. To run your application, perform these five procedures:

1. Set the TurboVNC password using `vncpasswd` (once).

   See "To Select a TurboVNC Password" on page 68..

2. Use Sun Grid Engine to perform these actions on your behalf:

   a. Select the graphics server.

   b. Select the graphics accelerator device.

   c. Start a TurboVNC session on the graphics server.

   See "To Start the TurboVNC Server Session" on page 68.

3. Start a TurboVNC viewer on your client (host). Additional viewers can be started by collaborators. This step is dependent on which TurboVNC viewer you use:
   - The Java based TurboVNC viewer software, which runs within a web browser on the client (simple).
   - The dedicated `vncviewer` client software component (better-performing).

   See "To Connect a TurboVNC Viewer to Your `RUN.vncserver` Session" on page 69.

4. Start your application within the TurboVNC session on the graphics server. Invoke the application using `vglrun`, so the application is under the control of VirtualGL.

   See "To Start a Graphics Application Within a TurboVNC Session" on page 70.

5. Eventually, terminate the TurboVNC server session (and all clients).

   See "To Terminate the `RUN.vncserver` Session" on page 71.

The following five sections detail these procedures, which identify differences specific to each TurboVNC viewer.

▼ To Select a TurboVNC Password

Before running the TurboVNC server for the first time, select a TurboVNC password. This password must differ from your login password.

- **Start** `vncpasswd`**:**

```
client% /opt/TurboVNC/bin/vncpasswd
Using password file /home/susieq/.vnc/passwd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
```

If `/opt/TurboVNC/bin` is in your `$PATH`, then you can start `vncpasswd`. The view-only password is an alternate password to be given to a collaborator you wish to enable to join your TurboVNC session. The collaborator can only view your session, not move the mouse, nor enter keyboard or mouse events. The TurboVNC password (and any view-only password) is used by all sessions started by this user using the same `$HOME` directory.

You can change the password before any session.

▼ To Start the TurboVNC Server Session

The `RUN.vncserver` script saves the TurboVNC server session connection information in files located in your `$HOME` directory. The script can only be used if the `$HOME` directory on the execution host is readable by the client host and only if these `$HOME` directories are the same. For example, the directories are NFS mounted. Consider that the `root` user has a different `$HOME` on each host and should not attempt to use the `RUN.vncserver` script.

- **Submit a job to Sun Grid Engine that starts the TurboVNC server session.**

  A script for this purpose is part of the Sun Shared Visualization 1.1 server installation.

```
client% qsub -now y $SGE_ROOT/graphics/RUN.vncserver
```

  Your grid can have a different script for this purpose, specific to your environment. You might also want additional `qsub` options, such as to specify the architecture of the graphics server Sun Grid Engine selects as your execution host. The `RUN.vncserver` script contains options for Sun Grid Engine to require a graphics device.

  The output (and any errors) from the `RUN.vncserver` script is appended to the file `$HOME/vncserver.log`. If your personal configuration files, such as `$HOME/.profile` or `$HOME/.cshrc` do not override the `$PATH` (or `csh $path`) established by the `RUN.vncserver` script, then `vglrun` (used to start a graphics application) is in your `$PATH`. (Design your configuration files to add paths to the `$PATH` that the files receive, and avoid replacing any path in `$PATH`.)

▼ To Connect a TurboVNC Viewer to Your `RUN.vncserver` Session

This procedure depends on whether your TurboVNC viewer is WebVNC or `vncviewer`. Either viewer works, as long as the graphics server and the client host share the same `$HOME` directory. `vncviewer` should perform significantly better.

- **Take one of the following actions:**
  - Connect a web browser to your TurboVNC session started with the `RUN.vncserver` script.

    The `RUN.vncserver` script creates files in your `$HOME` directory starting with `vnc_`. The file `$HOME/vnc_url` should redirect your browser to the execution server and port number for your TurboVNC session. If your web browser expands `$HOME`, you could simply enter (or select a bookmark for) `$HOME/vnc_url` or `file://$HOME/vnc_url`. If neither of these methods work, you can expand `$HOME` yourself and type `file://` and your home directory followed by `/vnc_url` (for example, `file:///home/susieq/vnc_url`). This action redirects your browser to the URL contained in your `vnc_url` file.

Note – The `$HOME` that contains the TurboVNC files is the `$HOME` on the execution host. This technique is convenient if the `$HOME` that contains the TurboVNC files is the same place as `$HOME` on the clients.

You can also view your `$HOME/vnc_url` file and use your browser to view the URL contained in that file (for example, `http://my_server:5802`).

The web page prompts you for the TurboVNC password and then enables you to view the TurboVNC session.

- Connect a TurboVNC viewer to your TurboVNC session started with the `RUN.vncserver` script.

  When you use the `RUN.vncserver` Sun Grid Engine job script to start your TurboVNC session, the script saves the graphics server name and port number in the file `$HOME/vnc_server` in a format useful to the TurboVNC viewer. You can start the TurboVNC viewer on your client (host) by appending `` `cat $HOME/vnc_server` `` as an option to your `vncviewer` starting:

```
client% /opt/TurboVNC/bin/vncviewer `cat $HOME/vnc_server`
```

  You can make a shell `alias` for this command.

Within this TurboVNC X session, you can create multiple terminals (shell windows) and start graphics applications. See "To Start a Graphics Application Within a TurboVNC Session" on page 70.

▼ To Start a Graphics Application Within a TurboVNC Session

- **Follow these guidelines when starting a graphics application.**

  Within the TurboVNC session, you might type commands to the graphics server's shell windows normally. However, when you are ready to run a graphics application, you must start VirtualGL's `vglrun` command manually. `vglrun` interposes between the application and the GLX library so that `vglrun` can read

back completed images from the graphics accelerator and pass the images to the TurboVNC session. The `vglrun` command can be in your `$PATH`. Otherwise, you need to use a full path to the `vglrun` command.

VirtualGL avoids compressing the graphics images VirtualGL gives to the TurboVNC session, because the TurboVNC session is on the same host and does not know how to decompress images. TurboVNC compresses images it sends to its viewer. Example startup from within a terminal in the TurboVNC session:

```
my_server% vglrun myprogram
```

That simple startup is sufficient when the `RUN.vncserver` job script was used to start the TurboVNC session, since the script sets `VGL_` environment variables into your environment.

In this second example, `vglrun` is not in the `$PATH`. The example uses an option to disable "spoiling".

```
my_server% /opt/VirtualGL/bin/vglrun -spoil myprogram
```

If you attempt to run an OpenGL application from within your TurboVNC session without remembering to use `vglrun` (but with `$DISPLAY` directing the application to your TurboVNC session), you might get an error message such as:

```
Xlib:  extension "GLX" missing on display "myserver:1.0"
```

▼ To Terminate the `RUN.vncserver` Session

You cannot just exit the viewer (that is, quit your web browser, leave the TurboVNC page, or exit the `vglviewer`) because the TurboVNC session continues. After you have saved your work, you must cause the TurboVNC session and all TurboVNC foreground processes to exit.

- **Take one of the following actions:**
  - Use the TurboVNC session's window manager logout procedure. (This is the simpler method.)

- From within the TurboVNC session or from any session to the same graphics server, issue a `kill` command to the graphics server host. The command must include the TurboVNC session's X display number, which `RUN.vncserver` saved in `$HOME/vnc_displayname`:

```
my_server% /opt/TurboVNC/bin/vncserver -kill `cat $HOME/vnc_displayname`
```

When the TurboVNC server session exits, `vglviewer` exits, but a web browser viewer prompts for a session password.

To list the X display numbers and process IDs of all TurboVNC server sessions that are currently running under your user account on this machine, type:

```
my_server% /opt/TurboVNC/bin/vncserver -list
```

CHAPTER 5

Advance Reservations

Advance Reservation (AR) is a feature of some queuing software systems but not yet present in Sun Grid Engine release 6.1. (If you are using a later release of Sun Grid Engine, check whether that version includes an Advance Reservation feature that can be used with Sun Shared Visualization software.) See the *Sun Shared Visualization 1.1.1 Release Notes* for more information.

Topics discussed in this chapter include:

Advance Reservation Overview

The Advance Reservation requirement is to schedule compute and visualization resources at a time when the computer resources and the persons to use the resources are both available. The Advance Reservation server makes this possible. Reservations must not be scheduled to conflict with each other (by oversubscribing available resources), nor to conflict with other Sun Grid Engine uses of the same resources.

A user can reserve specified resources at a given time, for a given duration. Once confirmed, the resources are available to that user's Sun Grid Engine jobs during that given reservation period. Jobs intended to run during the reservation period can be submitted to Sun Grid Engine (as with the Sun Grid Engine `qsub` command) right after the reservation is confirmed, or anytime before the end of the reserved period.

To use the optional Advance Reservation facility, the server (or client) requires the Java™ Runtime Environment (JRE™). The earliest version to support Advance Reservation is JRE version 1.5 (known as "Java 5").

Using the Advance Reservation Feature

To start the AR client, use the `bin/runar` script with either the `Reserve` (for command line) or `ReserveGUI` (for GUI) argument:

```
client% $SGE_ROOT/ar/bin/runar [arguments]
```

To simplify operations, you can create an alias containing the complete path to the `runar` script in a single command. For example, the following command creates an alias called `argui` that starts an AR GUI client:

```
client% alias argui '$SGE_ROOT/ar/bin/runar ReserveGUI'
```

Reserve AR Command-Line Client

The initial AR client is a command-line interface that connects with the AR server and handle reservations for the execution host. Any Sun Grid Engine user can run the AR client.

▼ To Start the AR Client

- **Type:**

```
client% $SGE_ROOT/ar/bin/runar Reserve [options]...
```

The following table describes some options for the `runar Reserve` command:

TABLE 5-1 `runar Reserve` Options

Option	Description
`-help`	Prints a description of the command-line options and formats.
`-serverHost` *ARserver*	Name of host that is running the Advance Reservation server software. The default *serverHost* is configured by the Advance Reservation server administrator.
`-host` *execHost* `-hostname` *execHost*	The execution host on which you want your reservation. Default is the first execution host configured on this AR server.

TABLE 5-1 `runar Reserve` Options *(Continued)*

Option	Description
`-a` *mmddHHMM*	Sets the reservation start date and time in the form of mmddHHMM (2-digit month, date, hour, and minute).
`-duration` *HH*`:`*MM*	Sets the reservation duration in the form of hours and minutes.
`-l` *resourceList*	Sets the reservation's resource list. Multiple resources can be comma-separated.
`-M` *EmailAddress*	Introduces your email address. Mail will be sent if the reservation is confirmed. See the `-m` option.
`-m` *specifier* `[`*minutes*`]`	Specifies when email is sent to the address given by the `-M` option, using a string made of any of these characters: • `r` *minutes* – A number of minutes before the beginning of the reservation. The number of minutes follows the specifier string. • `b` – at the beginning of the reservation • `e` – at the end of the reservation • `a` – if the reservation is aborted Example: **`-M myname@myhost -m ra 30`** This combination of options reminds 30 minutes before the reservation and also if the reservation is aborted.
`-N` *Name*	Name of reservation or project.
`-listreservations`	Lists all reservations for the user issuing the command.
`-deletereservation`	Deletes the key or queue for the reservation.
`-listresources`	Lists all resources that are requestable on any host.

For example:

```
runar Reserve -a 12250730 -duration 1:30:0 -l graphics=1,slots=1
```

In this example, there is a reservation for 7:30 AM on December 25, with a duration of one hour and 30 minutes. The resources required are one graphics resource and one slot.

Note – `slots` must be plural, even if the value is 1.

If successful, the command prints a `Confirmed` reservation, and issues a key or queue name. For example, `queue=shefali1163187511986`. The queue name is very important, as it is the handle used to submit jobs to the reservation's queue.

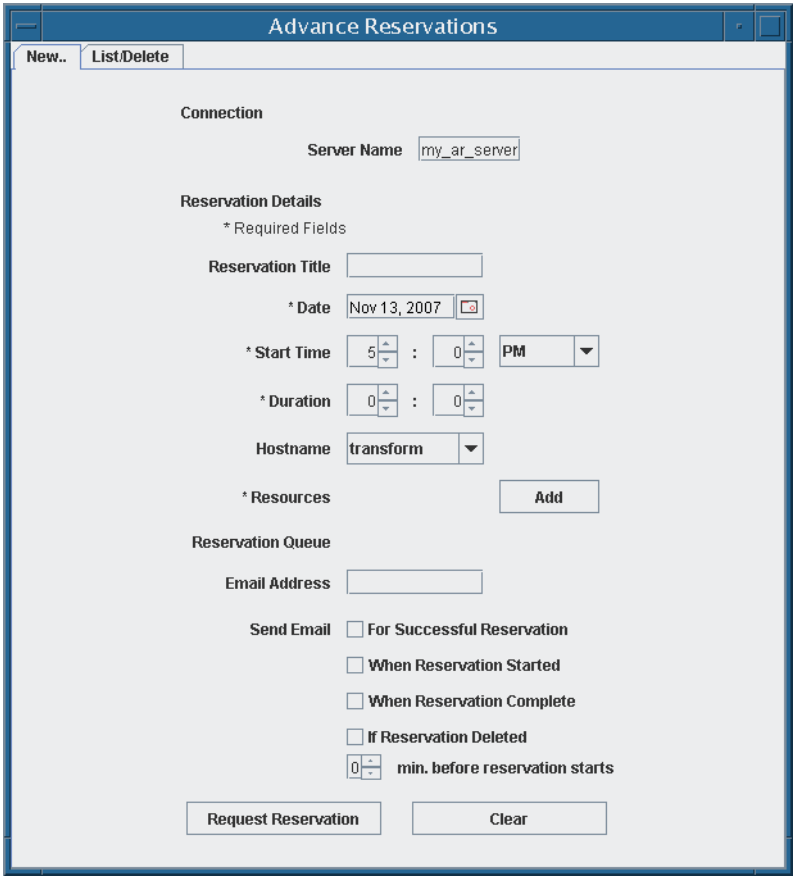
Reserve GUI Client

▼ To Start the AR GUI Client

- **Type:**

```
client% $SGE_ROOT/ar/bin/runar ReserveGUI [options]...
```

No *options* are normally required. The following GUI is displayed.



The New tab enables scheduling of a new reservation. TABLE 5-2 describes the purpose of each field under the New tab.

TABLE 5-2 Field Descriptions

Field	Description
Connection: Server Name	Host running the AR server, a name is normally obtained from the server's AR configuration file.
Reservation Title	A name for the particular reservation.
Date	Date for the reservation to start.
Start Time	Time at which the reservation starts.
Duration	The length of the reservation, in hours and minutes.
Hostname	The name of the execution host on which you want the reservation.
Resources	Sun Grid Engine resources to reserve.
Reservation Queue	Upon successful reservation submission and approval, this field has the assigned reservation queue name. This name is needed to submit jobs that use the reserved resources.
Email Address	Optional email address where a confirmation email can be sent. The mail message also contains the reservation's queue name.
Send Email	Specifies when email messages will be sent: • When the reservation is confirmed. • At the beginning of the reservation. • At the end of the reservation's duration. • If the reservation is deleted. • A number of minutes before the reservation starts.

▼ To See Pending Reservations

1. **Click on the List/Delete tab.**
2. **Type the user name where prompted, if not already provided.**

   Otherwise, reservations for all users will be displayed

.

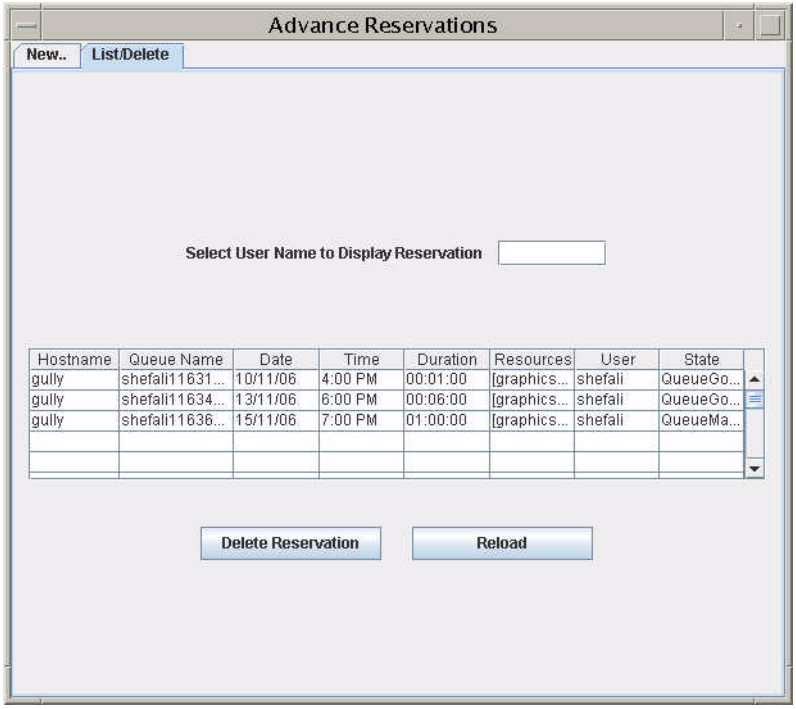


TABLE 5-3 describes the headings under the List/Delete tab.

TABLE 5-3 Advance Reservation List Heading Descriptions

Heading	Description
Hostname	The name of the reserved execution host.
Queue Name	The name that identifies the reservation and is needed to submit jobs that use the reserved resources. You can copy the queue name to the clipboard for pasting into another window or command line, such as when submitting a job to the reservation's queue.
Date	Date for the reservation to start.
Time	Time at which the reservation starts.
Duration	The length of the reservation, in hours, minutes, and seconds.
Resources	Sun Grid Engine resources to reserved.
User	The name of the user who requested the reservation.
State	Normally one of these values: QueueMade, (resources) Reserved, Started, Finished, or (resources) Returned.

▼ To Delete a Reservation

1. **Select the respective queue name from the list.**
2. **Click the Delete Reservation button.**

Submitting a Job to an Advance Reservation

One or more jobs can be submitted to an Advance Reservation queue, using the reservation's queue name. You can copy the queue name from a reservation confirmation (or from the list of reservations) and paste the name into the command line. For example:

```
qsub -q shefali1163187511986 -l graphics=1 my_gfx_script
```

If a reservation has only the default one slot, only one job can run at a time on that queue. However, additional jobs can be enqueued (without `-now` set to `y`).

When the reservation period (start time + duration) ends, any running job on the reservation's queue is stopped. A Sun Grid Engine administrator can alter the job to continue or to use a different queue. The administrator can configure a period after the reservation's finish time (default: 12 hours), which determines when the queue is deleted. Any job remaining on the queue after that time is killed.

APPENDIX **A**

VirtualGL Reference

This appendix provides information about VirtualGL options and environment variables that are most pertinent to the Sun Shared Visualization 1.1 software. There is also a summary of the VirtualGL GUI. Topics include:

Common `vglconnect` Scenarios

TABLE A-2 describes different scenarios for invoking `vglconnect`, the command to run the scenario, and respective comments.

TABLE A-1 Common `vglconnect` Scenarios

Scenario	Command	Comment
VGL Image Transport with X11 Forwarding	`vglconnect` *user@graphics-server*	• Starts `vglclient` (if it isn't running) • Opens `ssh` session to *graphics-server* with X tunnel.
X11 Forwarding, `ssh`-Encrypted VGL Image Transport	`vglconnect -s` *user@graphics-server*	`ssh` tunnel for VGL traffic can be useful through restrictive firewalls, and can create multilevel tunnels. You will probably be prompted twice for your password (on *graphics-server*).
Direct X11 Connection	`vglconnect -x` *user@graphics-server*	`client` host must allow remote (TCP) X connections.
From a Sun Ray	`vglconnect` is not used, use `ssh -X` to contact a remote graphics server.	See "Using VirtualGL From a Sun Ray Client" on page 34. `vglclient` is not needed when using the Sun Ray Image Transport.
Using VirtualGL with TurboVNC	`vglconnect` is not used, use `ssh` or `putty`.	See "Manually Using TurboVNC" on page 42. `vglclient` is not normally needed when using VNC.
After `vglclient -kill`, add the `-force` option to `vglconnect`	`vglconnect -force` *user@graphics-server*	This scenario forces `vglconnect` to start a `vglclient`, even if an earlier `vglclient` did not exit cleanly.
Granting SGE jobs access to the client's 2D X server	`vglconnect -k`	This command does not make a connection to the graphics server. The client host must also allow remote (TCP) X connections.

On Solaris, Linux, and Mac OS X hosts, if `/opt/VirtualGL/bin` is not in your path, you will have to type the full path:

```
/opt/VirtualGL/bin/vglconnect
```

On a Windows client, precede `vglconnect` commands with:

```
C> cd /d "c:\program files\virtualgl-version-build"
```

Common `vglrun` Scenarios

TABLE A-2 describes different scenarios for `vglrun`, the command to run the scenario, and respective comments. The *program* in the Command column can be either an application executable or a script; `$PATH` is searched if a pathname is not specified.

TABLE A-2 Common `vglrun` Scenarios

Scenario	Command	Comment
Remotely display a graphics application	`vglrun`	Uses the default image transport.
In VGL Image Transport, enable SSL encryption of the images that are sent to `vglclient`	`vglrun +s` *program*	This option has no effect with other image transports. The SSL port (default is 4243 for the client's first display) must be allowed through the client's firewall.
VGL Image Transport over Gigabit Networks	`vglrun -c rgb` *program*	Disable image compression. This option decreases server CPU consumption but drastically increases network bandwidth consumption.
Turn off frame spoiling	`vglrun -sp` *program*	This option is necessary to obtain accurate results from benchmark applications. Frame spoiling should be left on when running interactive applications.
In VGL Image Transport, set the JPEG quality to the value *q*	`vglrun -q` *q program*	Where *q* is a number between 1 and 100 (default is 95). This option has no effect in any other mode.
In VGL Image Transport, set the JPEG chrominance subsampling to *s*	`vglrun -samp` *s program*	Where *s* is one of the values from TABLE A-3. `1x` is the default for VirtualGL Image Transport. `16x` is the default for Sun Ray Image Transport.

A more thorough explanation of `vglrun` options and the corresponding environment variables is provided in "VirtualGL Options and Environment Variables" on page 86.

Chrominance Subsampling

When an image is compressed using JPEG or using Sun Ray DPCM and YUV compression, each pixel in the image is first converted from RGB (Red/Green/Blue) to YUV. A YUV pixel has three values that specify the overall brightness of the pixel (Y, or luminance) and the overall color of the pixel (U and V, or *chrominance*.)

Since the human eye is less sensitive to changes in color than to changes in brightness, the chrominance components for some of the pixels can be discarded without much noticeable loss in image quality. This technique, called chrominance subsampling, significantly reduces the size of the compressed image.

TABLE A-3 introduces available chrominance subsampling choices. For example, `8x` means to discard the chrominance components for 3 out of every 4 pixels horizontally and half the pixels vertically. (This option is also known as 4:1:0 or 4:2 subsampling.)

TABLE A-3 Chrominance Subsampling Characteristics

`-samp` Option	Subsampling	Performance	Image Bandwidth	Compression Artifacts
`1x`	4:4:4 (no subsampling)	Slowest	Most	None
`2x`	4:2:2 or 2:1 (discard 1/2 in X)	Medium slow	Medium high (about 20-25% less than 1x)	Some
`4x`	4:2:0 or 2:2 (discard 1/2 in X and Y)	Medium	Medium (about 35-40% less than 1x)	More
`8x`	4:1:0 or 4:2 (discard 3/4 in X and 1/2 in Y)	Medium fast	Medium low	Even more (available only with Sun Ray DPCM compression)
`16x`	4:4 (discard 3/4 in X and Y)	Fastest	Low	Still more (available only with Sun Ray DPCM compression)
`gray`	Discard all chrominance	Fastest	Least	Available only with JPEG compression

Narrow, aliased lines and other sharp features on a black background tend to produce very noticeable artifacts when chrominance subsampling is enabled. The `gray` option is useful when running applications (such as medical visualization applications) that are already generating grayscale images.

Gamma Correction

Gamma refers to the relationship between the intensity of light that your computer's monitor is instructed to display, and the intensity that it actually displays. The curve is an exponential curve of the form $Y = X^G$, where X is between 0 and 1. G is called the gamma of the monitor. PC monitors and TVs usually have a gamma of around 2.2.

Some of the mathematics involved in 3D rendering assumes a linear gamma (G = 1.0). Therefore, 3D applications will not display with mathematical correctness unless the pixels are *gamma corrected* to counterbalance the nonlinear response curve of the monitor. But some systems do not have any form of built-in gamma correction. Thus, the applications developed for such systems have usually been designed to display properly without gamma correction. Gamma correction involves passing pixels through a function of the form $X = W^{1/G}$, where G is the *gamma correction factor* and should be equal to the gamma of the monitor. So the final output is $Y = X^G = (W^{1/G})^G = W$. This equation describes a linear relationship between the intensity of the pixels drawn by the application and the intensity of the pixels displayed by the monitor.

You can enable, disable, or fine tune the gamma correction feature in VirtualGL by using the `VGL_GAMMA` environment variable and the `vglrun` options `-g`, `+g`, and `-gamma`. These operations are summarized in TABLE A-4.

Default Gamma Correction Behavior

By default, when VirtualGL remotely displays an OpenGL application, VirtualGL attempts to emulate the gamma correction behavior that would be observed when the same OpenGL application is displayed locally on the application server.

- On Solaris SPARC, the default is `VGL_GAMMA=1` or `vglrun +g` (that is, enable gamma correction using factor=2.2).

  When an OpenGL application is displayed locally on a Solaris SPARC host, the application's output is gamma corrected by default. Thus, VirtualGL will gamma correct the output of OpenGL applications that are displayed remotely from a Solaris SPARC host, unless you specify otherwise.

  If VirtualGL Image Transport is used and the client is a Solaris SPARC system, then VirtualGL will attempt to use a gamma corrected visual to perform gamma correction. (The gamma corrected visual feature is only available with Solaris SPARC graphics drivers.) When a gamma corrected visual is used, you can adjust the gamma correction factor by running `fbconfig` on the client machine.

  Otherwise (when a gamma corrected visual is not available or the client is not a Solaris SPARC system), VirtualGL uses its own internal gamma correction mechanism and the default gamma correction factor is 2.2.

- On Solaris x86 and Linux, the default is `VGL_GAMMA=0` or `vglrun -g` (that is, disable gamma correction).

  When an OpenGL application is displayed locally on a Solaris x86 or Linux host, the application's output will not be gamma corrected by default. Thus, VirtualGL will not gamma correct the output of OpenGL applications that are displayed remotely from a Solaris x86 or Linux host, unless you specify otherwise.

  Most non-SPARC clients provide a mechanism that allows you to gamma correct the entire screen, either through the graphics driver or the OS itself. However, these systems do not provide a means of gamma correcting an individual application (a capability of gamma corrected visuals).

VirtualGL Options and Environment Variables

`vglrun` is located:

- On a Solaris host at `/opt/VirtualGL/bin/` and at `/opt/SUNWvgl/bin`
- On a Linux host at `/opt/VirtualGL/bin/` and at `/usr/bin`

The syntax of `vglrun` is:

`vglrun [`*vglrun-options*`] [--]` *OpenGL-application* `[`*OpenGL-app-arguments*`]`

That is, any options for `vglrun` appear between `vglrun` and the name of the OpenGL application to start. If necessary, you can enter two hyphens to manually signify the end of `vglrun` options. Any options to the OpenGL application appear at the end of the line.

A more sophisticated example of the `vglrun` starting (which assumes `vglrun` is in your `$PATH`) is:

```
my_server% vglrun -g -d /dev/fbs/kfb0 my-program my-program-option my-program-argument
```

The most important `vglrun` option is controlled with `-d` (or with the `VGL_DISPLAY` environment variable). The value for the `-d` option is either an X display, or (on a Solaris SPARC host) the string `glp` or a GLP device name, such as `/dev/fbs/kfb0`,

as seen in the example. When an X display is used, the display should be on the graphics server with 3D graphics acceleration. (An optional *.screen* indicates the screen or device for the X server.) For example:

```
my_server% vglrun -g -d :0.1 my-program my-program-option my-program-argument
```

If your graphics server has only one graphics device, start a 3D X server on the device and enable that X server to you (or to everyone on the graphics server), as described in the "Configuration and Information" section of the *Sun Shared Visualization 1.1 Software Server Administration Guide*. After that, the default X display (`:0`) is fine. Ask your system administrator if you need to provide the display or device to `vglrun`.

Sun Grid Engine can set `VGL_DISPLAY` for you, when running a job that requests graphics resources. On a graphics server with only one graphics accelerator, `VGL_DISPLAY` can be set to the empty string `""`. The string is sufficient to cause VirtualGL to use the default X display (`:0.0`).

TABLE A-4, TABLE A-5, and TABLE A-6 list the VGL_ environment variables and equivalent vglrun options associated with general operation, Sun Ray Image Transport, and VGL Image Transport, respectively.

TABLE A-4 General VGL_ Environment Variables and vglrun Options

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_CLIENT = *client*:*display*	-client *client*:*display*	The host name, IP address, or X display where VirtualGL should send the VGL Image stream. VGL Image Transport uses a dedicated TCP/IP connection to transmit compressed images of an application's OpenGL rendering area from the application server to the client display. Thus, the server needs to know on which machine the VirtualGL client software is running, and which X display on that machine is used to draw the application's windows and other 2D GUI elements. The server's SSH_CLIENT or DISPLAY environment variable normally supplies this information to VirtualGL But if necessary, set VGL_CLIENT to the display where the application's GUI ends up. When using the VGL Image Transport, the display is the host name or IP address of the machine on which vglclient is running. When using the Sun Ray Image Transport, this display is the host name or IP address of the Sun Ray server. For example (using ksh/bash syntax): export VGL_CLIENT=my_client_machine:0.0	Set automatically by use of vlgconnect or ssh (as instructed in Chapter 3), or read from the DISPLAY environment variable (otherwise).

TABLE A-4 General VGL_ Environment Variables and vglrun Options *(Continued)*

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_COMPRESS=proxy VGL_COMPRESS=jpeg VGL_COMPRESS=rgb VGL_COMPRESS=sr VGL_COMPRESS=srrgb VGL_COMPRESS=sryuv	-c [proxy | jpeg | rgb | sr | srrgb]	Selects image transport and image compression type. Each value selects a specific image transport: • proxy selects X11 Image Transport. • jpeg or rgb selects VGL Image Transport. • sr, srrgb, or sryuv selects Sun Ray Image Transport. proxy = Sends images uncompressed using the X11 Image Transport. This option is useful when displaying to a local X server or X proxy such as TurboVNC. jpeg = Compresses images using JPEG and sends using the VGL Image Transport. This option is used with vglclient when displaying to a remote X server. rgb = Encodes images as uncompressed RGB and sends using the VGL Image Transport. This option is useful when displaying to a remote X server or X proxy across a very fast network. sr* modes below all require that the proprietary Sun Ray plug-in be installed on the VirtualGL server, that the client is a Sun Ray ultrathin client, and that the Sun Ray Image Transport was active when VirtualGL started. The VGL default is to use sr compression in this case. VGL will fall back to using proxy compression if, for any given frame, it is unable to send the frame using the Sun Ray Image Transport. This situation could occur if, for example, the Sun Ray client is on a network that is not visible to the VirtualGL server. sr = Compresses images using the Sun Ray default compression method (DPCM) and sends using the Sun Ray Image Transport. srrgb = Encodes images as uncompressed RGB and sends using the Sun Ray Image Transport. sryuv = Encodes images as uncompressed YUV and sends using the Sun Ray Image Transport.	If the X server is a Sun Ray server and the Sun Ray plug-in is installed on the VirtualGL server, use sr (if possible) and proxy (otherwise). If the X server is not a Sun Ray server, use proxy if DISPLAY begins with : or with unix: (otherwise, use jpeg).

TABLE A-4 General VGL_ Environment Variables and vglrun Options *(Continued)*

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_DISPLAY	-d *display-or-GLP-device*	The X display or GLP device to use for 3D rendering. You may set VGL_DISPLAY to the name of any graphics-accelerated X display (such as :0.1). This functionality could be used, for instance, to support many application instances on a multipipe graphics server. GLP mode (Solaris SPARC only) – Setting this option to GLP or glp enables GLP mode and selects the first available GLP device for rendering. You can also set this option to the pathname of a specific GLP device (for example, /dev/fbs/jfb0 or /dev/fbs/kfb0). GLP is a special feature of the Sun SPARC OpenGL library that enables an application to render into Pbuffers on a jfb or kfb graphics card, even if there is no 3D X server running on that graphics card. Sun Grid Engine can set VGL_DISPLAY for you, when running a job that requests graphics resources. On a host with only one graphics device, Sun Grid Engine might set VGL_DISPLAY to the null string (""). Consequently, the default X server (:0.0) is used.	:0
VGL_FPS=*f*	-fps *f*	Limits the client-server frame rate to *f* frames per second, where *f* is a floating point number > 0.0. This option can be used, for instance, as a crude way to control network bandwidth or CPU usage in multiuser environments where those resources are constrained. This option prevents the VGL and X11 Image Transports from sending frames at a rate faster than the specified limit. If frame spoiling is disabled, then this option effectively limits the server's graphics rendering frame rate as well.	No limit
VGL_GAMMA=1	+g	Enables gamma correction with default settings. This option enables gamma correction using the best available method. The client's gamma-corrected X visual is used, if available. Otherwise, VirtualGL performs gamma correction internally using a default gamma correction factor of 2.22. This option emulates the default behavior of OpenGL applications running locally on SPARC systems.	This is the default on Solaris SPARC application servers (gamma correction enabled, factor=2.2)

TABLE A-4 General VGL_ Environment Variables and vglrun Options *(Continued)*

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_GAMMA=0	-g	Disables gamma correction. This option tells VGL not to use gamma-corrected visuals, even if they are available on the 2D X server, and disables VGL's internal gamma correction system as well. This option emulates the default behavior of OpenGL applications running locally on Linux or Solaris x86 machines.	This is the default on Solaris x86 and Linux application servers (gamma correction disabled, factor=1.0)
VGL_GAMMA=*factor*	-gamma *factor*	Enables VGL's internal gamma-correction system with the gamma-correction factor specified. If VGL_GAMMA is set to an arbitrary floating-point value (a decimal point should always be used), then VirtualGL performs gamma correction internally using the specified value as the gamma-correction factor. You can also specify a negative value to apply a *de-gamma* function. Specifying a gamma correction factor of G (where G < 0) is equivalent to specifying a gamma correction factor of -1/G.	2.2 for VGL_GAMMA=1 1.0 for VGL_GAMMA=0
VGL_GUI		Specifies key sequence to pop-up VirtualGL's dynamic control GUI. See "VirtualGL GUI for Quality and Performance Tradeoff" on page 95.	Shift-Ctrl-F9
VGL_INTERFRAME=0 VGL_INTERFRAME=1		Disables and enables interframe comparison. Interframe comparison only sends tiles of the image that have changes since the previous frame. This option has no effect in X11 Image Transport.	Comparison enabled.
VGL_LOG		Redirects the console output from VirtualGL to a log file. Setting this environment to the pathname of a log file on the VirtualGL server causes the VirtualGL faker to redirect all of its messages to the specified log file rather than to stderr. Output content includes any profiling and trace output.	Print all messages to stderr.
VGL_PROFILE=0 VGL_PROFILE=1	-profile +profile	Disables and enables profiling output. If enabled, this option causes the VirtualGL faker to continuously benchmark itself and periodically print out the throughput of reading back, compressing, and sending pixels to the client. See also VGL_SPOIL.	Profiling disabled.

TABLE A-4 General VGL_ Environment Variables and vglrun Options *(Continued)*

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_NPROCS=*n*	-np *n*	Selects the number of CPUs to use for multithreaded compression. The VGL Image Transport can divide the task of compressing each frame among multiple server CPUs. This behavior might speed up the overall throughput in rare circumstances where the server CPUs are significantly slower than the client CPUs. VirtualGL will not allow more than four processors total to be used for compression. Nor will VirtualGL allow you to set this parameter to a value greater than the number of processors in the system.	1
VGL_SPOIL=0 VGL_SPOIL=1	-spoil +spoil	Disables and enables frame spoiling. By default, VirtualGL drops frames so the newest rendered frame, perhaps based on the most recent mouse movement, is presented to the user as soon as possible. This functionality should produce the best results with interactive applications. However, turn off frame spoiling when running benchmarks or other noninteractive applications. Turning off frame spoiling forces one frame to be read back and sent on each buffer swap, thus enabling benchmarks to accurately measure the frame rate of the entire VirtualGL pipeline. Note that in X proxy environments (for example, TurboVNC), the application's frame rate might still not reflect the frame rate presented to the user. Disabling frame spoiling also prevents noninteractive applications from wasting graphics resources by rendering frames that are never seen. With frame spoiling turned off, the rendering pipeline behaves as if the pipeline's fill-rate is limited to about 30 or 40 Megapixels/second, the maximum throughput of the VirtualGL system on current CPUs.	Spoiling enabled.

TABLE A-4 General VGL_ Environment Variables and vglrun Options *(Continued)*

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_STEREO=left VGL_STEREO=right VGL_STEREO=quad VGL_STEREO=rc	-st [left | right | quad | rc]	Specifies the delivery method for stereo images (when an application renders a stereo frame). left – Sends only the left eye buffer. right – Sends only the right eye buffer. quad – Attempts to use quad-buffered stereo, which will result in a pair of images being sent to the VirtualGL client on every frame. Quad-buffered stereo requires the VGL Image Transport, and it also requires that client support OpenGL and have a 3D accelerator that supports stereo rendering. If quad-buffered stereo is not available, either because the client or the Image Transport does not support it, then falls back to using anaglyphic stereo. rc – Uses Red/Cyan (anaglyphic) stereo, even if quad-buffered stereo is available.	quad – use quad-buffered stereo if available, use anaglyphic stereo otherwise.
VGL_SUBSAMP=1x VGL_SUBSAMP=2x VGL_SUBSAMP=4x VGL_SUBSAMP=8x VGL_SUBSAMP=16x VGL_SUBSAMP=gray	-samp [1x|2x|4x|8x|16x|grey]	Subsamples chrominance (color) to improve performance at the expense of quality. Since the human eye is less sensitive to changes in color than it is to changes in brightness, the chrominance components for some of the pixels can be discarded without much noticeable loss in image quality. 1x – Full YUV color resolution (4:4:4). 2x – full resolution in Y, subsamples U and V by 2 in X (4:2:2). 4x – Subsamples U and V by 2 in X and Y (4:2:0). 8x – Subsamples U and V by 4 in X, 2 in Y (4:1:0). 16x – Subsamples U and V by 4 in X and Y (4:4). gray – Discards all chrominance (color) components. Applies to DPCM and YUV enccoding (Sun Ray Transport) and JPEG encoding (VGL image tranpsort). See TABLE A-3 on page 84.	1x (full color resolution).
VGL_TRACE=0 VGL_TRACE=1	-tr +tr	Disables or enables tracing. When tracing is enabled, VirtualGL logs all calls to the GLX and X11 functions it is intercepting. VirtualGL also logs the arguments, return values, and execution times for those functions. This behavior is useful when diagnosing interaction problems between VirtualGL and a particular OpenGL application.	Disabled.

TABLE A-4 General VGL_ Environment Variables and vglrun Options *(Continued)*

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_VERBOSE=0 VGL_VERBOSE=1	-v +v	Disables or enables verbose VirtualGL messages. When verbose mode is enabled, VirtualGL reveals some of its decisions, such as which code path it is using to compress images, which type of X11 drawing it is using, and so on. This behavior can be helpful when diagnosing performance problems.	Disabled.

TABLE A-5 VGL_ Environment Variables and vglrun Options for Sun Ray Image Transport

Environment Variable Name	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_ZOOM_X=2		Downsamples YUV an extra factor of 2 in X.	Disabled.
VGL_ZOOM_Y=2		Downsamples YUV an extra factor of 2 in Y.	Disabled.
VGL_PROGRESSIVE=0 VGL_PROGRESSIVE=1	-prog +prog	*This option is no longer recommended or supported.* Disables and enables sending of a Lossless version of the image if a newer image is not provided. The transmission is interrupted if a new image arrives.	Disabled.

TABLE A-6 VGL_ Environment Variables and vglrun Options for VGL Image Transport

Environment Variable Name and Values	vglrun Command-Line Override	vglrun Option Description	Default Value
VGL_QUAL=[1-100]	-q [1-100]	An integer between 1 and 100 (inclusive). This setting enables you to specify the quality of the JPEG compression. Lower is faster but also grainier. The default setting should produce visually lossless performance. 100 is is mathematically lossless except for round-off errors.	95
VGL_SSL=0 VGL_SSL=1	-s +s	Disables or enables SSL encryption (of VGL images). Enabling this option causes the VGL Image Transport to be tunneled through the secure socket layer (SSL).	Disabled.

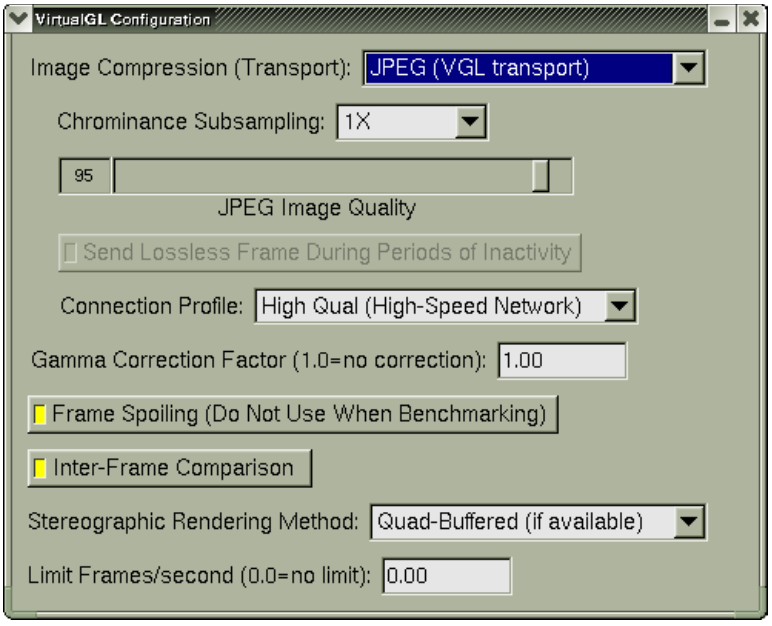
Passing a configuration option as an argument to vglrun effectively overrides the environment variable setting corresponding to that configuration option.

For more information about `vglrun` options, start `vglrun -help` or consult the *VirtualGL User's Guide*. See "Related Documentation" on page xvii.

VirtualGL GUI for Quality and Performance Tradeoff

VirtualGL has a small graphical user interface (GUI) to enable you to dynamically control the VirtualGL compression and encoding. See FIGURE A-1. This functionality enables you to reconfigure visual quality and performance on-the-fly.

FIGURE A-1 VirtualGL's Configuration Dialog (Showing LAN Defaults)



▼ To Start the VirtualGL GUI

- **Press the Control-Shift-F9 keys.**

  That is, hold the Control and Shift keys, and press the F9 function key. You can use the `VGL_GUI` environment variable to change the key combination if, for example, an application is already using that key sequence.

  For example:

```
my_server% setenv VGL_GUI CTRL-F9
```

  The command changes the key sequence to the Control-F9 keys for graphics programs started with `vglrun` from that shell.

Using the VirtualGL GUI

You can use this dialog to adjust various image compression and display parameters in VirtualGL. Changes are reflected immediately in the application. TABLE A-7 describes each of the setting fields for the GUI. Many of these descriptions refer to

static settings described in tables in "VirtualGL Options and Environment Variables" on page 86. The GUI overrides those static settings from an environment variable or option to `vglrun`.

TABLE A-7 VirtualGL GUI Field Descriptions

Field	Description
Image Compression (Transport)	Selects the image compression and transport technique. Inappropriate choices are grayed out (for example, Sun Ray transport choices are unavailable on other clients). None (X11 Transport) – Equivalent to `VGL_COMPRESS=proxy`. This option can be activated at any time, regardless of which transport was active when VirtualGL started. JPEG (VGL Transport) – Equivalent to `VGL_COMPRESS=jpeg`. This option is only available if the VGL Image Transport was active when the application started. RGB (VGL Transport) – Equivalent to `VGL_COMPRESS=rgb`. This option is only available if the VGL Image Transport was active when the application started. DPCM (Sun Ray Transport) – Equivalent to `VGL_COMPRESS=sr`. This option is only available if the Sun Ray Image Transport was active when the application started. RGB (Sun Ray Transport) – Equivalent to setting `VGL_COMPRESS=srrgb`. This option is only available if the Sun Ray Image Transport was active when the application started. YUV (Sun Ray Transport) – Equivalent to setting `VGL_COMPRESS=sryuv`. This option is only available if the Sun Ray Image Transport was active when the application started.
Chrominance Subsampling	Selects the color compression (quality and performance tradeoff) when using JPEG or Sun Ray DPCM compression. This option overrides `VGL_SUBSAMP`, described in TABLE A-4 on page 88 and in TABLE A-3 on page 84.
JPEG Quality	This slider enables you to set a percentage quality when using JPEG compression. A higher percentage means better quality, at the expense of reduced bandwidth. Changing this setting overrides `VGL_QUAL`.

TABLE A-7 VirtualGL GUI Field Descriptions *(Continued)*

Field	Description
Connection Profile	This drop-down menu is active only if the VGL Image Transport was active when the application started. All choices set the image compression type to JPEG (VGL Transport). It has the following options in release 1.1.1: • Low Qual (Low-Bandwidth Network) – Also sets the Chrominance Subsampling to 4X, and sets the JPEG Image Quality to 30. • Medium Qual – Also sets the Chrominance Subsampling to 2X, and sets the JPEG Image Quality to 80. • High Qual (High-Bandwidth Network) – Also sets the Chrominance Subsampling to 1X, and sets the JPEG Image Quality to 95. These settings are VirtualGL's current defaults.
Gamma Correction Factor	This floating-point field is the equivalent of setting `VGL_GAMMA`. If using a gamma-corrected visual (SPARC clients only), then this field has no effect. Otherwise, this field enables VirtualGL's internal gamma-correction system with the specified gamma-correction factor.
Frame Spoiling	Toggles between frame spoiling on (enabled) or off (disabled). Changing this setting overrides the value of `VGL_SPOIL`.
Interframe Comparison	Toggles interframe comparison on and off. Changing this setting overrides the value of `VGL_INTERFRAME`.
Stereographic Rendering Method	Drop-down menu overrides the value of `VGL_STEREO`.
Limit Frames/second	This floating-point field overrides `VGL_FPS`.

Note – VirtualGL monitors the application's X event loop to determine whenever a particular key sequence has been pressed. If an application is not monitoring key press events in its X event loop, then the VirtualGL configuration dialog might not pop up at all.

vglclient options

The following table lists options honored by vglclient when using the VGL Image Transport. Note that vglclient is normally started implicitly by vglconnect.

TABLE A-8 vglclient Options for VGL Image Transport

Environment Variable Name and Values	vglclient Command-Line Override	vglclient Option Description	Default Value
VGLCLIENT_DRAWMODE= ogl | x11	-gl -x	Specifies the method used to draw pixels into the application window. Options use OpenGL or X11.	ogl for Solaris SPARC systems with 3D accelerators, x11 otherwise
VGLCLIENT_LISTEN= sslonly | nossl	-sslonly -nossl	Accepts only SSL connections or only unencrypted connections from the VirtualGL server.	Accept both
VGL_PROFILE=0 VGL_PROFILE=1		Disables or enables profiling output. If profiling output is enabled, then VirtualGL will continuously benchmark itself and periodically print out the throughput of various stages in its image pipelines. See also VGL_SPOIL.	Disabled
VGL_VERBOSE=0 VGL_VERBOSE=1		Disables or enables verbose VirtualGL messages. When verbose mode is enabled, VirtualGL reveals some of its decisions, such as which code path it is using to decompress images, which type of X11 drawing it is using, and so on. This behavior can be helpful when diagnosing performance problems.	Disabled

Advanced OpenGL Features

This section discusses VirtualGL features that support these advanced OpenGL Features:

- "Stereographic Rendering" on page 100 (rendering image pairs intended for left and right eyes)
- "Transparent Overlays" on page 102 (rendering an image on top of another rendered image, so the top image can be easily redrawn without requiring a redraw of the underlay)

Stereographic Rendering

Stereographic rendering is a feature of OpenGL that creates separate rendering buffers for the left and right eyes and allows the application to render a different image into each buffer. How the stereo images are subsequently displayed depends on the specifics of the 3D hardware and the user's environment.

VirtualGL can support stereographic applications in two ways:

Stereographic Mode	Description	Requires
Quad-Buffered Stereo	Sends the stereo image pairs to the client to be displayed in stereo by the client's 3D graphics card	Stereo-capable 3D graphics hardware on server and client.
Anaglyphic Stereo	Combines each stereo image pair into a single *anaglyph* that can be viewed with widely-available red/cyan 3D glasses	Stereo-capable 3D graphics hardware on server (3D graphics hardware is not needed on the client)

You can select a specific stereo mode by setting the `VGL_STEREO` environment variable, by using the `-st` argument with `vglrun` (see TABLE A-4 on page 88), or by using the VirtualGL GUI (see "VirtualGL GUI for Quality and Performance Tradeoff" on page 95).

Quad-Buffered Stereo

The name *quad-buffered* stereo is derived from OpenGL using four buffers (left front, right front, left back, and right back) that support stereographic rendering with double buffering. 3D graphics cards with quad-buffered stereo capabilities typically provide a synchronization signal that can control active stereo 3D glasses of various types. Some 3D graphics cards support *passive stereo*, which displays the buffers for the left and right eyes to different monitor outputs. The outputs might be projected onto the same screen through polarized filters.

VirtualGL supports true quad-buffered stereo by rendering stereo images on the server and sending image pairs across the network. The image pairs are displayed by a 3D graphics card on the client.

Quad-Buffered Stereo Requirements

In most cases, the VirtualGL and TurboVNC clients use only 2D drawing commands, so the client host does not required a 3D graphics card. But displaying quad-buffered stereo images requires that the client host have a 3D graphics card. Since

the 3D graphics card is only being used to display images, it does not need to be a high-end card of that type. In most cases, the least expensive 3D graphics card that has stereo capabilities will work fine in a VirtualGL client.

Also, the server must have a 3D graphics card that supports stereo, since this is the only way that a stereo Pbuffer is provided to VirtualGL.

When an application tries to render something in stereo, VirtualGL will use quad-buffered stereo rendering if all of the following characteristics are true:

- VGL Image Transport is being used.
- Anaglyphic stereo is not explicitly requested (for example, by `VGL_STEREO`, by `vglrun`'s `-st` option, or by VirtualGL's GUI).
- The client supports OpenGL (Exceed 3D is required for Windows clients).
- The client has stereo rendering capabilities.
- The server has stereo rendering capabilties.

If any of these conditions is not true, then VirtualGL falls back to using anaglyphic stereo (as described in "Anaglyphic Stereo" on page 101). You usually need to explicitly enable stereo in the graphics driver configuration for both the client and server machines. "Verifying Advanced Feature Support" on page 109 describes how to verify that stereo visuals are available on both the client and server.

In quad-buffered mode, VirtualGL reads back both eye buffers on the server, then sends the contents as a pair of compressed images (one for each eye) to the VirtualGL client. The VirtualGL client then decompresses both images and draws them as a single stereo frame to the client machine's X display using `glDrawPixels()`. Because of this process, enabling quad-buffered stereo in VirtualGL typically decreases performance by 50 percent or more, and twice the network bandwidth is used to maintain the same frame rate.

Anaglyphic Stereo

Anaglyphic stereo is the type of stereographic display used for old 3D movies. This method usually relies on a set of 3D glasses consisting of red transparency film over the left eye and cyan transparency film over the right eye. To generate a 3D anaglyph, the red color data from the left eye buffer is combined with the green and blue color data from the right eye buffer, enabling a single monographic image to contain stereo data. Within the capabilities of VirtualGL, an anaglyphic image is the same as a monographic image. Therefore, anaglyphic stereo images can be sent by any image transport to any type of client, regardless of the client's capabilities.

VirtualGL falls back to using anaglyphic stereo when VirtualGL detects that an application has rendered something in stereo but quad-buffered stereo is not available. Quad-buffered stereo might be unavailable because the client doesn't support it or because the image transport being used is not GL Image Transport.

Anaglyphic stereo provides an inexpensive and simple way to view stereographic applications in X proxies (including TurboVNC) and on clients that do not support quad-buffered stereo. Additionally, anaglyphic stereo performs much faster than quad-buffered stereo, since quad-buffered stereo sends twice as much data to the client.

Transparent Overlays

Transparent overlays render an overlay image on top of an underlay rendered image. You can easily erase or redraw the overlay image without requiring the underlay to be redrawn. Transparent overlays have requirements and restrictions similar to those for quad-buffered stereo. For an application to use this feature, transparent overlay features must be provided by the client host's 3D graphics card and OpenGL.

When an application performs OpenGL rendering to the transparent overlay, VirtualGL completely bypasses its own GLX faker. Instead, VirtualGL uses indirect OpenGL rendering to render to the transparent overlay on the client's graphics card. The underlay is still, as always, rendered on the server host.

Use of overlays is becoming more and more infrequent. When they are used, it is typically only for drawing small, simple, static shapes and text. Usually it is faster to send the overlay geometry over to the client rather than to render it as an image and send the image.

As with stereo functions, sometimes overlays must be explicitly enabled in the client graphics card's configuration. Unlike stereo requirements, overlays need to be supported and enabled only on the client machine.

Indexed color (8-bit) overlays have been tested and are known to work with VirtualGL. Use glxinfo (see Troubleshooting below) To verify whether your client's X display supports overlays and if overlays are enabled, use glxinfo as described in "To Verify Client Features" on page 109. In Exceed 3D, make sure that the Overlay Support option is checked in the Exceed 3D and GLX applet.

Overlays do not work with X proxies, including TurboVNC. VirtualGL must be displaying to a real X server to use overlays.

Troubleshooting Common Errors

This section describes common user errors and how to avoid, detect, or recover from the errors.

vglconnect and ssh Issues

- In recent Solaris OS releases (such as Solaris 10), the Secure Shell daemon (sshd) has the default of not allowing remote logins as user root. This default is configured by the PermitRootLogin entry in the ssh daemon's configuration file, /etc/ssh/sshd_config. To avoid this problem, login as a user other than root when running ssh or vglconnect (which uses ssh). Become root only when necessary by using the su command.

VirtualGL Issues

- If the VirtualGL Client is not started, an attempt to start an application under VirtualGL control reports Connection refused:

```
[VGL] Could not connect to VGL client. Make sure the VGL client is running and
[VGL]    that either the DISPLAY or VGL_CLIENT environment variable points to
[VGL]    the machine on which it is running.
[VGL] rrsocket.cpp--
[VGL] 224: Connection refused
```

  This message is probably caused by using a program such as ssh to log into the graphics server, rather than using vglconnect, which starts vglclient implicitly. Another possible cause is that the vglclient process exited. In that situation, a new vglclient needs to be started using the vglconnect -force option. Check the VirtualGL client log file for possible errors.

- If VirtualGL is not compressing images, VirtualGL can draw directly to the client's X server without connecting to the VirtualGL client software, but this behavior does not perform as well. There is not an automatic fallback when vglclient is not found. The Connection refused message also appears if the client is a Sun Ray and the Sun Ray plug-in for VirtualGL (package SUNWvglsr) has not been installed on the graphics server (unless no compression was requested).
- If the DISPLAY environment variable is not set on the graphics server host, then an attempt to start an application under VirtualGL control reports something similar to: XOpenDisplay: Error opening display. Or if access was not granted properly, the application might print messages such as:

```
Xlib: connection to "client:0.0" refused by server
Xlib: Client is not authorized to connect to Server
myapplication: XOpenDisplay: Error opening display.
```

- If the graphics server's X display or GLP graphics device is not properly configured, you do not have permission to access the display or device. VirtualGL reports an error such as:

```
[VGL] Could not open display (null).
```

  In this case, `null` indicates that the VirtualGL display or device specification was `""` (a null string, which is the VirtualGL default, meaning the graphics server's `:0.0` display). The name of the display or device name that VirtualGL failed to open or access is printed.

  This message might be caused by one of the following configuration or usage errors:

  - The "3D X server" running on the graphics server has not been configured to allow access to VirtualGL users. This configuration procedure is described in Chapter 4 of the *Sun Shared Visualization 1.1 Server Administration Guide*.
  - The current user account is not in the `vglusers` group. To do this procedure, see Chapter 4 of the *Sun Shared Visualization 1.1 Server Administration Guide*.
  - There is no "3D X server" running on the graphics server providing access to the graphics accelerator.
  - The graphics server is configured for use only with GLP, but `vglrun` was not provided with the `-d glp` argument.

- If the graphics server does not offer a true color (RGB) X visual, VirtualGL might print:

```
Error: couldn't get an RGB, Double-buffered visual
```

  The same error can occur if the client's "2D X server" does not offer a 24-bit true color (RGB) visual. Some Linux systems are configured for only 16-bit visuals. In this case, the system must be reconfigured for 24-bit true color visuals.

  Another possible reason that the application can't open a usable visual is that the graphics server's 3D graphics card does not support OpenGL pixel buffers (Pbuffers). The graphics server might not have the proper driver installed for that 3D graphics card. On Linux, you probably need to use the driver supplied by the 3D graphics card vendor instead of the driver that was included with the operating system. For example, use the `nvidia` driver supplied by the 3D graphics card vendor rather than an `nv` driver from another source.

- If you neglect to start `vglrun`, the graphics application can use the GLX remote graphics technique described in "Sun Shared Visualization 1.1 Introduction" on page 1 if `$DISPLAY` directs the application to your client's X server and the server supports the GLX (OpenGL for X) graphics extension. GLX can be far slower than VirtualGL, especially for large graphics models.

- Error message such as the following might indicate that the `ssh` server daemon running on the graphics server does not have X11 forwarding enabled:
  - `Could not open display`
  - `Client is not authorized to connect to server`
  - `Connection refused by server`

  To configure X11 forwarding, see the *Sun Shared Visualization 1.1 Server Administration Guide*.

`vglrun` Issues With Set-UID Programs and Scripts

When a Solaris or Linux application (binary executable or script) must set the user or group ID when it runs (enabled with a UNIX file system's `setuid` and `setgid` permission bits), that application might generate errors when it runs under VirtualGL. The error message depends on the operating system.

- On Solaris, the message might be:

```
warning: /opt/SUNWvgl/lib/librrfaker.so: open failed: illegal insecure pathname
```

- On Linux, the message might be:

```
ERROR: ld.so: object 'librrfaker.so' from LD_PRELOAD cannot be preloaded:
ignored.
```

This error occurs because the VirtualGL faker library is preloaded into every executable that the script launches. When invoking an executable that is `setuid` or `setgid`, the operating system refuses to load virtualGL into those executables, because you are attempting to preload a library (VirtualGL) that the operating system does not consider secure into a process that needs to be secure.

- The Solaris software allows preloading into a `setuid` or `setgid` process only if the library is in a directory that the software recognizes as containing only secure libraries.
- Linux versions allow preloading into a `setuid` or `setgid` process a library only if it is in the standard search directories (such as `/lib`) that are also `set-user-ID`.

Alternatives to Work With This Restriction

A system administrator can configure the operating system to consider the VirtualGL library to be secure, as is described in the Configuration chapter of the *Sun Shared Visualization 1.1 Software Server Administration Guide*.

If the application is a script, a more secure method is to edit the application script and have it run `vglrun` only for the executables that you want to run in the VirtualGL environment. Here are various ways to do this:

- `Use vglrun's -32` and `-64` options on OpenSolaris or Solaris to provide control for launching scripts:

`vglrun` **Option**	**Description**
`vglrun -32`	Preloads VirtualGL only into 32-bit executables.
`vglrun -64`	Preloads VirtualGL only into 64-bit executables.

  Here is an example of using these options. The script calls a binary needing `setuid` that is a 32-bit execuatable. However, the graphics application is a 64-bit executable. In this situation, you can use `vglrun -64` to launch the application script. The result is that the 32-bit setuid binary will not attempt to preload VirtualGL's faker library.

- You can edit the application script (or create an alternative script) so the script postpones use of `vglrun` until `vglrun` invokes the actual graphics application. For example, your original script (called `my_script`) is as follows:

```
#!/bin/sh
some_setuid_binary
some_application_binary
```

  Rather than running this with `vglrun my_script`, you can create a similar script (called `my_vgl_script`) as follows:

```
#!/bin/sh
some_setuid_binary
/opt/VirtualGL/bin/vglrun some_application_binary
```

  Invoke `my_vgl_script` directly (that is, do not enter `vglrun my_vgl_script`). The result is that this script does not attempt to preload VirtualGL into `some_setuid_binary`, but will preload VirtualGL into `some_application_binary`, as you wanted. Unfortunately, this method does allow the user invoking `my_vgl_script` to provide `vglrun` options on the command line. But the next technique does.

- A variation for editing an application script allows the invoking user to provide `vglrun` options on the command line, as long as the user is not counting on preloading any libraries other than VirtualGL. This method depends on saving and restoring environment variables in the modified script. The environment variables manipulated depend on the server's Operating System.
  - On OpenSolaris or Solaris, a modified application script should save and restore the `LD_PRELOAD_32` and `LD_PRELOAD_64` variables:

```
#!/bin/sh
LD_PRELOAD_32_SAVE=$LD_PRELOAD_32
LD_PRELOAD_64_SAVE=$LD_PRELOAD_64
LD_PRELOAD_32=
LD_PRELOAD_64=
export LD_PRELOAD_32 LD_PRELOAD_64

some_setuid_executable

LD_PRELOAD_32=$LD_PRELOAD_32_SAVE
LD_PRELOAD_64=$LD_PRELOAD_64_SAVE
export LD_PRELOAD_32 LD_PRELOAD_64

some_application_executable
```

  - On Linux, the modified application script should save and restore the `LD_PRELOAD` variable:

```
#!/bin/sh
LD_PRELOAD_SAVE=$LD_PRELOAD
LD_PRELOAD=
export LD_PRELOAD

some_setuid_executable

LD_PRELOAD=$LD_PRELOAD_SAVE
export LD_PRELOAD

some_application_executable
```

vglclient Messages (Normally in the Log for vglconnect)

vglclient prints messages as graphics server applications connect and disconnect. vglconnect normally redirects vglclient's output to a log file whose name is printed by vglconnect

After a connection, the vglconnect log file normally contains connection and disconnection messages from vglclient, such as:

```
++ Connection from 10.4.22.34.
Error receiving data from server.  Server may have disconnected.
   (this is normal if the application exited.)
rrsocket.cpp-- 394: Incomplete receive
-- Disconnecting 10.4.22.34
```

Depending on how the application exits, messages might be printed before the Disconnecting message, as the Incomplete receive message shown here. If the application was supposed to exit, these messages are of no concern. But these messages can be used to help analyze unexpected behavior.

vis_report Reporting Script

Sun Shared Visualization 1.1 software includes a reporting script, /opt/SUNWvrpt/bin/vis_report, that is helpful in debugging product installation, configuration, and usage problems.

Support engineers can use the script output to troubleshoot problematic behavior. Use of the script is dependent upon the situation when the problem occurs:

- **VirtualGL** – In most cases, run the script on the graphics server, ideally from the client host that attempted to use VirtualGL and as the user attempting to use VirtualGL.
- **TurboVNC** – If you are using a TurboVNC session, run the script from within that session. The $DISPLAY and other environment variables will be set the same as when the problem occurred.
- **Advance Reservation** – If the Advance Reservation facility for Sun Grid Engine is involved, also run the script on the host running the Advance Reservation server, as the owner of the $SGE_ROOT/ar/config directory.

Attach the output of the script to an email of your problem. Describe what you were trying to do, what you expected to happen, and what actually occurred. Send the email to your service provider or to `Shared-Viz-Support@Sun.com`.

Verifying Advanced Feature Support

VirtualGL includes a modified version of `glxinfo` that can be used to determine whether or not the client and server have stereo, overlay, or Pseudocolor visuals enabled.

▼ To Verify Quad-Buffered Stererographics on the Server

- **Run one of the following command sequences on the VirtualGL server to determine whether the server has a suitable visual for stereographic rendering.**
  - On a Solaris server (using GLP):

```
% /opt/VirtualGL/bin/glxinfo -d {glp_device} -v
```

  - On a Linux or Solaris server (not using GLP):

```
% xauth merge /etc/opt/VirtualGL/vgl_xauth_key
% /opt/VirtualGL/bin/glxinfo -display :0 -c -v
```

In the output, one or more of the visuals should list `stereo=1` and should list `Pbuffer` as one of the Drawable Types.

▼ To Verify Client Features

- **Run the following command sequence on the VirtualGL server:**

```
% /opt/VirtualGL/bin/glxinfo -v
```

Examine the output to determine whether the X display on the client has a suitable visual to support stereographic rendering, transparent overlays, or Pseudocolor

- In order to use stereo, one or more of the visuals should list `stereo=1`.
- In order to use transparent overlays, one or more of the visuals should list `level=1`, should list a Transparent Index (nontransparent visuals will say "Opaque" instead), and should have a class of PseudoColor.
- In order to use PseudoColor (indexed) rendering, one of the visuals should have a class of PseudoColor.

GLX Spheres Test Program

The GLX Spheres test program (`glxspheres`) is found in `/opt/VirtualGL/bin`. You can use this program to verify that VirtualGL has been configured and invoked properly.

This program supports the options shown in TABLE A-9.

TABLE A-9 `glxspheres` Options

Option	Description
`-h`	Help – Prints a summary of options and exit.
`-c`	Uses color index rendering. The default is true color (RGB).
`-fs`	Full Screen. Size the window to consume the entire X screen, without any window manager ornamentation. (Press `q` or Escape keys to exit the program.)
`-i`	Interactive mode. Frames advance only when the mouse is clicked or dragged. Continuously dragging the mouse in the window should produce a steady frame rate. This frame rate is a reasonable model of the frame rate that you can achieve when running interactive applications in VirtualGL.
`-m`	Uses immediate mode rendering. (The default is display list rendering for maximum performance. Many applications cannot use display lists, because the geometry they are rendering is dynamic. So this option models how such applications might perform when displayed remotely without VirtualGL.)
`-o`	Overlays. Renders text, a moving crosshair cursor, and a block of pixels to an 8-bit transparent overlay while animating the spheres in the underlay. Also changes the overlay's color map periodically.
`-s`	Uses stereographic rendering initially. (Later, stereo can be switched on and off in the application using VirtualGL's Configuration dialog. See "VirtualGL GUI for Quality and Performance Tradeoff" on page 95.)

APPENDIX B

TurboVNC Reference

This appendix provides basic reference information about TurboVNC. Topics include:

For instructions in using the TurboVNC server, see "Manually Using TurboVNC" on page 42. The TurboVNC commands are not normally in your `PATH`. Either add their location `/opt/TurboVNC/bin` to your `PATH` or enter full pathnames to the following commands.

Common TurboVNC Scenarios

TurboVNC Server Scenarios

TABLE B-1 describes different scenarios for the TurboVNC server, the `vncserver` command, and respective comments.

TABLE B-1 Common TurboVNC Server Scenarios

Scenario	Command	Comment
Start a TurboVNC session with default settings.	`vncserver`	The X display number of a TurboVNC session is printed out whenever you start the session.

TABLE B-1 Common TurboVNC Server Scenarios *(Continued)*

Scenario	Command	Comment
Start a TurboVNC session with a given virtual desktop size.	`vncserver -geometry` *w* x *h*	Where the desktop is *w* x *h* pixels in size. Default is 1240x900 pixels.
List all your TurboVNC sessions.	`vncserver -list`	Lists all the TurboVNC sessions of the current user on this host.
Kill the TurboVNC session of X display number *display*.	`vncserver -kill :`*display*	TurboVNC sessions can only be killed by the user that started the session.

Upon startup, the TurboVNC server session uses the user's VNC startup file. If the file does not exist, the TurboVNC session creates one. This startup file is named `$HOME/.vnc/xstartup.turbovnc` on release 1.1.1 and `$HOME/.vnc/xstartup` on previous releases. The xstartup file that TurboVNC creates uses operating system specific techniques to launch the user's most recently used window manager.

TurboVNC Viewer Scenarios

On a Windows host, start a TurboVNC viewer by selecting `TurboVNC Viewer` in the TurboVNC Start Menu group. A small GUI (shown in FIGURE 3-1 on page 48) appears to allow selection of image encoding protocol and related parameters. These are further described in"TurboVNC Image Encoding Protocols and Dynamic Quality/Performance Tradeoff" on page 116

TABLE B-2 describes different scenarios for starting a TurboVNC viewer from a command line. Options that would be used on UNIX and on Windows are provided.

TABLE B-2 Common TurboVNC Viewer Scenarios

Scenario	UNIX, Mac OS X, and Windows Commands	Comment
Connect to the VNC server session running on machine *host* that has an X display number of *display*.	`vncviewer` *host*`[:`*display*`]`	Note the single colon, as is standard for an X display name.
Similar to previous scenario, but do not allow others to view or share your session.	`vncviewer -noshared` *host*`[:`*display*`]` `vncviewer /noshared` *host*`[:`*display*`]`	The default is to allow any user who correctly enters your VNC password to view your session.
Set the JPEG quality to *q*.	`vncviewer -quality` *q* *host*`[:`*display*`]` `vncviewer /quality` *q* *host*`[:`*display*`]`	Where *q* is a number between 1 and 100 (default is 95). Once connected, you can change this dynamically using the F8 menu.

TABLE B-2 Common TurboVNC Viewer Scenarios *(Continued)*

Scenario	UNIX, Mac OS X, and Windows Commands	Comment
Set the JPEG chrominance subsampling to *s*.	`vncviewer -samp` *s* *host*`[:`*display*`]` `vncviewer /samp` *s* *host*`[:`*display*`]`	Where *s* is `1x` for no subsampling (4:4:4), `2x` for 4:2:2 subsampling, `4x` for 4:2:0 subsampling, or `gray` for no chominance. Default is `1x`. Once connected, you can change this setting dynamically using the F8 menu. See "Chrominance Subsampling" on page 84 for more information.
Improve performance, at the expense of image quality.	`vncviewer -medqual` *host*`[:`*display*`]`	Use Medium Quality connection profile (on UNIX or Mac OS X).
Minimize bandwidth consumption at the expense of image quality.	`vncviewer -lowqual` *host*`[:`*display*`]`	Use Low Quality connection profile (on UNIX or Mac OS X).
Connect to the VNC server session running on machine *host* and listening on port *port*.	`vncviewer` *host*`::`*port*	Note the double colons.

TurboVNC Image Encoding Protocols and Dynamic Quality/Performance Tradeoff

TurboVNC allows user control to balance the (often conflicting) goals of high image quality and high performance. First we describe the options that control TurboVNC's compression of images in TABLE B-3. Then we describe predefined option combinations in TABLE B-4 on page 117.

TABLE B-3 TurboVNC Quality and Performance Options

TurboVNC Option Name	`vncviewer` Command-Line Override	`vncviewer` Option Description	Default Value
Allow JPEG compression	`-nojpeg`	By default, TurboVNC will use JPEG compression for all image tiles with more than 24 unique colors and paletted encoding for all other image tiles. With `-nojpeg`, TurboVNC will select between paletted or raw encoding depending on the size of the image tile and its color count.	Allow JPEG
JPEG image quality (an integer between 1 and 100, inclusive)	`-quality [1-100]`	This setting enables you to specify the quality of the JPEG compression. Lower is faster but also grainier. The default setting should produce visually lossless performance. 100 is is mathematically lossless except for round-off errors. If image quality is of paramount concern, consider setting the JPEG quality to 100 or using RGB encoding.	95
JPEG chrominance subsampling	`-samp [1x|2x|4x|8x|16x|grey]`	Subsamples chrominance (color) to improve performance at the expense of quality. `1x` – Full YUV color resolution (4:4:4). `2x` – Full resolution in Y, subsamples U and V by 2 in X (4:2:2). `4x` – Subsamples U and V by 2 in X and Y (4:2:0). `8x` – Subsamples U and V by 4 in X, 2 in Y (4:1:0). `16x` – Subsamples U and V by 4 in X and Y (4:4). gray – Discards all chrominance (color) components.	1x
`Zlib` compression level		If Zlib compression is enabled, then paletted and raw-encoded image tiles will be compressed with Zlib prior to transmission. This decreases network bandwidth usage at the expense of increased server CPU usage. If JPEG compression is enabled, then Zlib compression is always used with non-JPEG image tiles.	`1`

TABLE B-4 lists the most useful combinations of the compression options above. These are called the "Image Encoding Protocol" in release 1.1.1. In previous releases, these were known as Connection Profiles (with a slightly different set of options). "Tight" refers to the image encoding used by TightVNC, on which TurboVNC is based.

TABLE B-4 Useful Combinations of TurboVNC Quality and Performance Options

Image Encoding Protocol Name and UNIX Command-line option	Allow JPEG?	JPEG Image Quality	JPEG Chrominance Subsampling	Zlib Compression Level	Protocol Description
Tight + Perceptually Lossless JPEG (UNIX default)	Yes	95	1x	N/A	This protocol should be perceptually lossless; that is, any image compression artifacts it produces should be imperceptible to the human eye under most viewing conditions. This protocol requires a great deal of network bandwidth, however, and is generally not recommended except on 50 Megabit/second and faster networks.
Tight + Medium Quality JPEG `-medqual`	Yes	80	2x	N/A	For image tiles with a high number of unique colors, this protocol produces some minor, but generally not very noticeable, image compression artifacts. All else being equal, this protocol typically uses about twice the network bandwidth of the "Low Quality" protocol and about half the bandwidth of the "Perceptually Lossless" protocol, making it appropriate for medium-speed networks such as 10 Megabit ethernet.
Tight + Low Quality JPEG `-lowqual`	Yes	30	4x	N/A	For image tiles with a high number of unique colors, this protocol produces very noticeable image compression artifacts. However, it performs optimally on low-bandwidth connections. If image quality is more critical than performance, then use one of the other encoding protocols or take advantage of the "Lossless Refresh" feature. See "Lossless Refresh" on page 120.
Lossless Tight `-lossless`	No	N/A	N/A	0	This protocol uses paletted encoding for image tiles with a low number of unique colors but otherwise does not perform any image compression at all. It is thus suitable only for gigabit and faster networks. This protocol uses significantly less CPU time than any of the JPEG-based protocols.
Lossless Tight + Zlib `-losslesswan`	No	N/A	N/A	1	This protocol uses paletted encoding for image tiles with a low number of unique colors and raw encoding for image tiles with a high number of unique colors. It compresses all image tiles using Zlib with compression level 1. For certain types of applications (CAD applications, in particular), this protocol uses less network bandwidth than the "Perceptually Lossless" protocol, but it also uses significantly more CPU time on the server than any of the JPEG-based protocols.

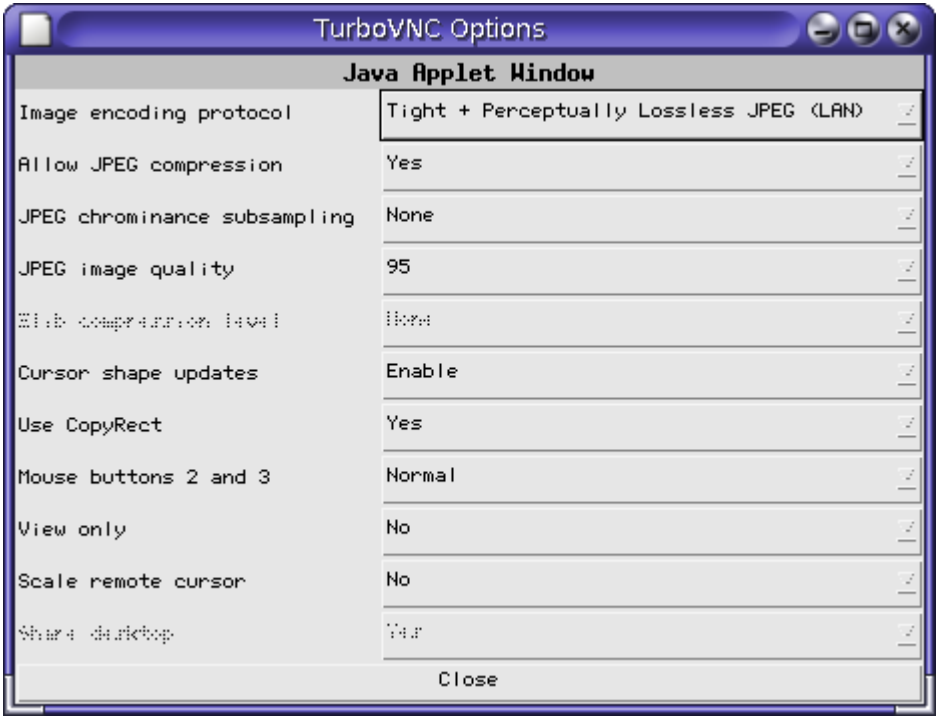
▼ To Select the Image Encoding Protocol

This selection can be made statically for your viewer session or changed dynamically "on-the-fly" as you use your viewer session. Selecting the Image Encoding depends on your viewer.

- For the Java viewer, click on the Options button at the top of the "TurboVNC Connection" dialog box. Then select the desired image encoding protocol (see FIGURE B-1).

  For dynamic control of the protocol, after connecting to your server, click the Options button at the top of the browser window to obtain the same dialog.

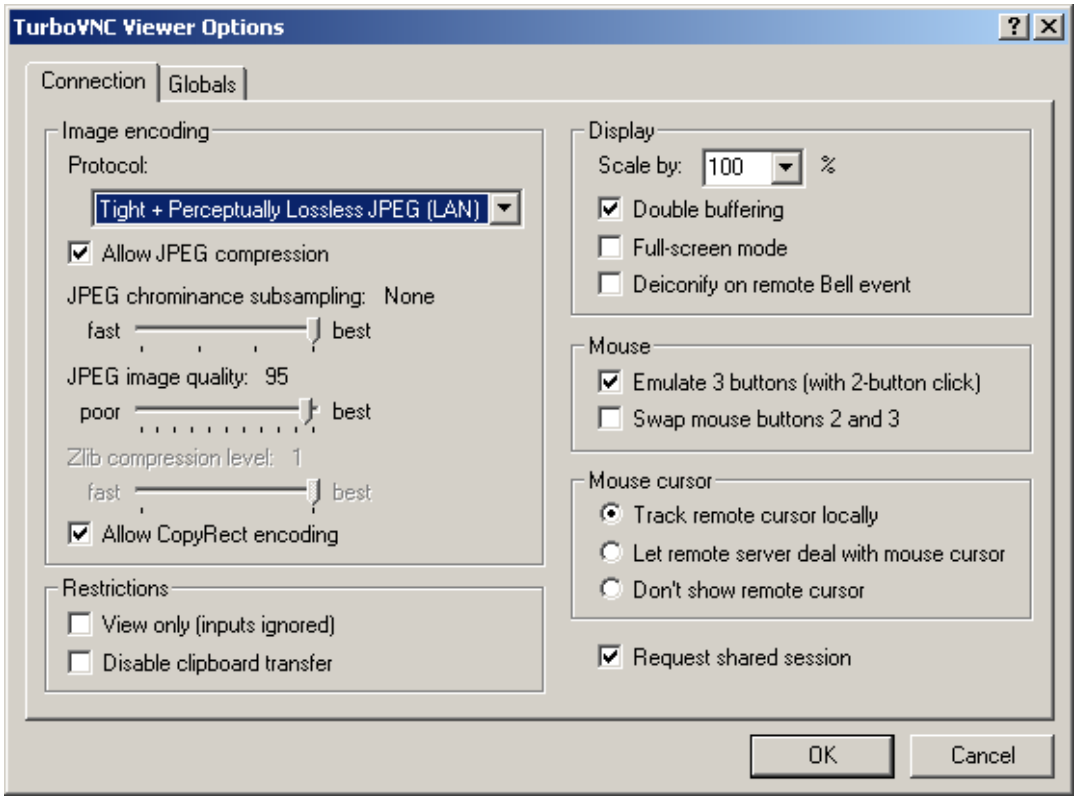
FIGURE B-1 WebVNC Options Dialog



- In the Windows TurboVNC Viewer, click on the Options button in the TurboVNC Connection dialog. Then select the Image delivery and other attributes, as shown in FIGURE B-2. (If you do not choose settings, the Windows viewer stores the last settings that were used with a particular VNC server *host*:*display* and tries to use those settings whenever you connect to the same display again.)

  For dynamic control of the protocol, after connecting to the server, click on the Connection Options button in the toolbar to obtain the same dialog.

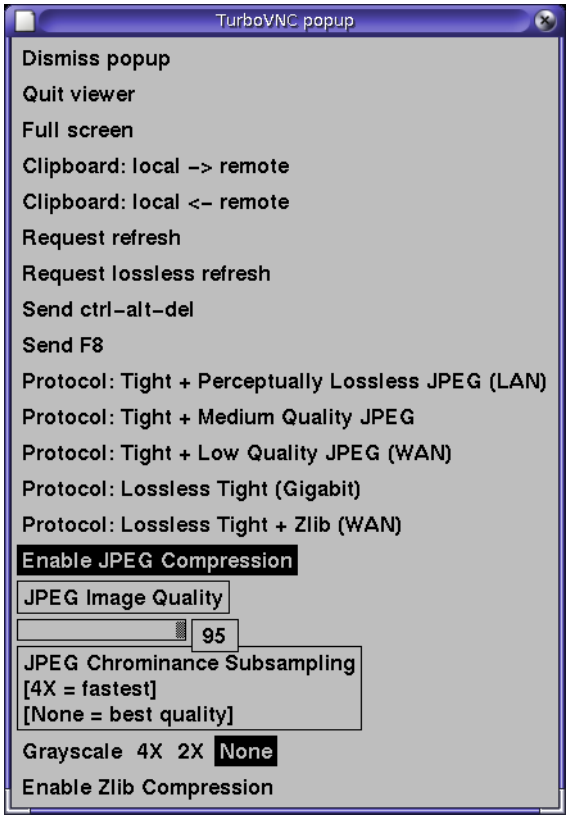
FIGURE B-2 TurboVNC Viewer Options Dialog on a Windows Client



- In the Solaris, Linux, and Mac OS X TurboVNC Viewer, the "Tight + Perceptually Lossless JPEG" encoding is the default. You can use `vncviewer`'s command-line options, given in TABLE B-4 on page 117 to select a nondefault image encoding protocol from that table.

  For dynamic control of the protocol, you can press the F8 key after connecting to display a menu (see FIGURE B-3) from which you can dynamically control the image encoding protocol and parameters.

FIGURE B-3 TurboVNC's Configuration Dialog (Defaults for Perceptually Lossless JPEG Are Shown)



Lossless Refresh

TurboVNC can optionally compress images losslessly, but this mode does not perform well except on extremely fast networks. Another option for quality-critical applications is the *Lossless Refresh* feature. Lossless Refresh causes the server to send a mathematically lossless copy of the current screen to the viewer. So, for instance, you can rotate, pan, or zoom an object in your application using a very lossy quality setting. Then, when you are ready to interpret or analyze the object closely, you can request a lossless refresh of the screen.

▼ To Perform a Lossless Refresh

- **Take one of the following actions, depending on your viewer:**

- In the Solaris, Linux, or Mac OS X TurboVNC Viewer, select Lossless Refresh from the F8 pop-up menu.
- On a Windows TurboVNC viewer, either press Ctrl-Alt-Shift-L or click on the Lossless Refresh toolbar icon.
- In the Java TurboVNC Viewer, click the Lossless Refresh button at the top of the browser window.

Troubleshooting Common TurboVNC Session Startup Errors

X Font Server Issues

On some systems, when you start the TurboVNC server session, this message appears:

```
Couldn't start Xvnc; trying default font path.
Please set correct fontPath in the vncserver script
or make sure that the X Font Server (xfs) is running.
```

Usually after this message is displayed, the TurboVNC session starts, but without all the fonts that would be available if the font server were running.

This message is normally caused by the X font server not running. This issue is unlikely to occur with TurboVNC 0.5 (in Sun Shared Visualization 1.1.1) or later releases, because it attempts to find the fonts it needs without depending on the font server. If you encounter font server issues, you can, as superuser, configure the X font server to start automatically.

▼ To Configure the X Font Server to Start Automatically

- **As superuser, enter the command appropriate to the operating system.**
  - On a Solaris 10 graphics server, enter:

```
# svcadm enable stfsloader xfs
```

- On a Solaris graphics server running a version earlier than Solaris 10, enter:

```
# fsadmin -e
```

- On a Linux system, to configure the X font server to start automatically with each boot, enter:

```
# chkconfig xfs on
```

  On many Linux systems, you also can start the X font server immediately. Enter:

```
# /etc/init.d/xfs restart
```

X Authentication Issues

TurboVNC relies on `xauth`, the X authentication control program. Your script that starts up `vncserver` might not find `xauth` unless it is included in your `PATH`. When `xauth` is not found, the TurboVNC session might fail to start or might start an X server that cannot be connected to by your X clients (such as windows or a window manager).

To avoid this problem, make sure that `xauth` is included in your `PATH`. Also, when you invoke the TurboVNC server, avoid unintentionally invoking a vncserver command that shares the X directory. This might happen if the X directories are listed prior to `/opt/TurboVNC/bin` in your PATH.

For more information on this issue, see CR 6710919 at `http://sunsolve.sun.com`.

`xstartup` Issues

Upon startup, the TurboVNC server session uses the user's VNC startup file. If the file does not exist, the TurboVNC session creates one. This startup file is named `$HOME/.vnc/xstartup.turbovnc` on release 1.1.1. On previous releases, it was named `$HOME/.vnc/xstartup` and could contain contents from a different VNC server, which was incompatible with TurboVNC.

The startup file that TurboVNC creates uses operating system specific techniques to launch the user's most recently used window manager. If necessary, you can edit the startup file created by TurboVNC.

APPENDIX C

Sun Grid Engine Reference

This appendix provides basic information about the Sun Grid Engine commands and options. More thorough information is available in the Sun Grid Engine documentation. See "Related Documentation" on page xvii.

Topics in this section include:

Accessing the Sun Grid Engine Environment

To access Sun Grid Engine, the client host NFS mounts the Sun Grid Engine installation. Your client host should mount Sun Grid Engine so that you can use the same `$SGE_ROOT` as the NFS server does. (The default is `/gridware/sge`.)

▼ To Access the Sun Grid Engine Environment

1. **Test the accessibility of the `$SGE_ROOT` directory from your client host:**

```
# ls /net/nfsserverhostname/gridware
```

where `/gridware` is the base directory of your `$SGE_ROOT`.

2. **From your client host, access the NFS server's** `$SGE_ROOT` **as the client's own** `$SGE_ROOT` **using** `/etc/vfstab`**,** `/etc/fstab`**, or automount.**

Note – Client hosts must not mount the NFS server with `nosuid` option, since `setuid` is needed by `rlogin` and `rsh`.

- For Solaris automounting:

a. Add the following line to the `/etc/auto_direct` **file:**

```
/gridware  -rw,suid,bg,hard,noquota,intr nfsserverhostname:/gridware
```

where `/gridware` is the base directory of your `$SGE_ROOT`.

b. Restart the automounter:

- For the Solaris 10 (or later) OS

```
# svcadm -v restart autofs
```

- For earlier Solaris releases:

```
# /etc/init.d/autofs stop ; /etc/init.d/autofs start
```

Note – The easiest method to automount every file system from the NFS server is to create a symbolic link. For example:
ln -s /net/*nfsserverhostname***/$SGE_ROOT $SGE_ROOT**
However, you must ensure that such a mount allows `suid` access.

- For Linux mounting:

a. Add the following line to the `/etc/fstab` **file:,**

```
nfsserverhostname:/gridware    /gridware  nfs        auto,suid,bg,intr          0 0
```

b. Type these two commands:

```
# mkdir /gridware
# mount /gridware
```

where `/gridware` is the base directory of your `$SGE_ROOT`.

Note – If you use NIS to resolve host names, add the server's name to the `/etc/hosts` file and ensure that `files` appears in the `hosts` entry in `/etc/nsswitch.conf`

3. **If your grid installation requires it, copy the server's `sge_qmaster` line from the server's `/etc/services` file into your client's.**

   This step is not needed if the SGE settings files set the `SGE_QMASTER_PORT` environment variable. See "Setting Up the Sun Grid Engine Environment Variables" on page 125.

Setting Up the Sun Grid Engine Environment Variables

▼ To Set Up the Sun Grid Engine Environment Variables

- **Set up the Sun Grid Engine environment variables:**
  - `tcsh` and `csh` users set up environment variables using:

```
submit_host% source /gridware/sge/default/common/settings.csh
```

    Substitute `/gridware/sge` with your value for `$SGE_ROOT`.

  - `sh`, `bash`, and `ksh` users use:

```
$ . /gridware/sge/default/common/settings.sh
```

    Substitute `/gridware/sge` with your value for `$SGE_ROOT`.

Note – These commands add `$SGE_ROOT/bin/$ARCH` to `$path`, add `$SGE_ROOT/man` to `$MANPATH`, set `$SGE_ROOT`, and if needed, set `$SGE_CELL` (probably `default`). These commands probably also set your `SGE_QMASTER_PORT` environment variable.

You might want to insert a command like these in your login configuration file, probably subject to a test that the settings file exists (is readable).

Basic Sun Grid Engine Commands

TABLE C-1 provides a brief description of the basic Sun Grid Engine commands.

TABLE C-1 Basic Sun Grid Engine Commands

Command	Description
`qmon &`	Starts a graphical user interface (GUI) for displaying the Sun Grid Engine state and for submitting jobs. A Sun Grid Engine administrator can also use the GUI to alter the state of Sun Grid Engine.
`qstat`	Shows jobs you have submitted, yet are not complete.
`qstat -f`	Shows available queues and execution hosts, the architecture (operating system and processor type), the current state (`au` means unavailable), all running jobs, and other information.
`qsub`	Submits a job for future execution. This job might need to wait until necessary resources are available. Job output is saved in files.
`qrsh`	Submits an interactive job. If the job cannot start immediately, you are told to try again later. The job is not queued. Job output goes to the invoking window.

If SGE commands such as `qstat` are still not found after setting up the environment, have your system administrator verify that the NFS server contains binaries for your client's architecture (operating system and processor, as output by SGE's `arch` command at `$SGE_ROOT/util/arch`). For example, if `arch` prints `sol-amd64`, then `$SGE_ROOT/bin` should contain a subdirectory named `sol-amd64`.

`qsub` job output is redirected to a file (and `qrsh` output is not). By default, this file is `$HOME/`*JobName*`.o`*JobId*. Any error output is likewise saved in the error file, which defaults to `$HOME/`*JobName*`.e`*JobId*. Other differences between `qsub` and `qrsh` are presented in TABLE C-3. Both `qsub` and `qrsh` require an absolute (starting with `/`) or relative path to the program or script to be submitted. `$PATH` will not be searched to locate `qsub` or `qrsh`.

qsub and qrsh Commands

The qsub command starts batch jobs at a later time. The qrsh command runs jobs interactively.

Some Common qsub and qrsh Options

TABLE C-2 provides command options common to both qsub and qrsh.

TABLE C-2 Common qsub and qrsh Options

Option	Description
-v *variable*	Introduces environment variables whose values should be copied from the current shell to the job. You can also use -v *variable=value* to assign the value that should be saved with the job for that variable.
-q *queue-name*	Enables you to demand that your job execute on a particular queue. Using wildcards such as "*@myserver", you can demand any queue on a certain host without specifying which queue. Quoting is needed to pass the wildcard characters to qsub, rather than having the characters expanded by your interactive shell.
-l *resource=value*[,*resource=value*]...	Specifies Sun Grid Engine job resource attributes.
graphics=1	Allocates use of a graphics accelerator.
arch=*string*	Where *string* identifies the processor and operating system. For example: sol-sparc64 Solaris on SPARC (64-bit) sol-sparc Solaris on SPARC (32-bit) sol-amd64 Solaris on x64 (64-bit) sol-x86 Solaris on x86 (32-bit) lx24-amd64 Linux (2.4 or 2.6 kernel) on x64 (64-bit) lx24-x86 Linux (2.4 or 2.6 kernel) on x86 (32-bit) Wildcarding is supported, if quoted to keep the submit shell from expanding the wildcards. For example: "sol-sparc*" Solaris on SPARC (32-bit or 64-bit) "*-x86" Solaris or Linux on x86 (32-bit) "lx24-*" Linux (2.4 or 2.6 kernel, 32-bit or 64-bit)

TABLE C-2 Common `qsub` and `qrsh` Options

Option		Description
-	`h_rt=`*hour*`:`*minute*`:`*seconds* `s_rt=`*hour*`:`*minute*`:`*seconds*	Hard runtime limit. After the specified hard runtime limit, Sun Grid Engine aborts the job using the `SIGKILL` signal. If the similar `s_rt` soft limit is reached, Sun Grid Engine warns the job by sending the job the `SIGUSR1` signal. This behavior is effective only if the job catches and handles that warning signal. Jobs that do not specify an elapsed time limit inherit a system default. The default is necessary for the Advance Reservation system to assure resource availability.

Different Default Behavior of `qsub` and `qrsh`

Though the `qsub` and `qrsh` commands start jobs, their respective default behavior is different. TABLE C-3 presents the differences in `qsub`'s and `qrsh`'s defaults for certain options.

TABLE C-3 Differences in `qsub` and `qrsh` Command Options

Option	Mnemonic	qsub Default (batch job)	qrsh Default (interactive)	Behavior
`-now [yn]`	`now`	`n`	`y`	If the job cannot run immediately: `y` = Submission fails. `n` = Spool the job for later.
`-b [yn]`	`binary`	`n`	`y`	`n` = Target script file is copied into the job and scanned for `#$` options (job default functions). `y` = Neither of these events happen.
`-w [ewnv]`	`warn`	`n`	`e`	
`e`	`error`			Fail submit if job cannot run.
`w`	`warning`			Print message if job cannot run.
`n`	`none`			Enqueue syntactically valid jobs.
`v`	`verify`			Explain any reason job cannot run.

Note – Use the `-w` option of `qsub` or `qrsh` to obtain more information about why Sun Grid Engine cannot schedule a job to run.

Example Sun Grid Engine Job Script

The following example job script starts `/opt/VirtualGL/bin/glxspheres` on a Solaris or Linux graphics server. This script is a simplified version of `$SGE_ROOT/graphics/RUN.glxspheres`. Italicized text in this listing provides commentary, but is not part of the job script itself.

```
#!/bin/sh                                          This script is interpreted by the Bourne shell, sh.
#
# The name of my job:
#$ -N glxspheres
#
# The interpreter SGE must use:
#$ -S /bin/sh                                      Sun Grid Engine always uses sh to interpret this script.
#
# Join stdout and stderr:
#$ -j y
#
# This job needs a graphics device:
#$ -l gfx=1                                        # Allocate a graphics resource to this job.
#
# Specify that these environment variables are to be sent to SGE with the job:
#$ -v DISPLAY
#$ -v VGL_CLIENT
#$ -v VGL_GAMMA
#$ -v VGL_GLLIB
#$ -v VGL_SPOIL
#$ -v VGL_X11LIB
#$ -v SSH_CLIENT
# If these variables are not set before qsub/qrsh is invoked,
# then the job will find these variables set, but with a null string value ("").
#
# Script can run on what systems?
# Solaris (SPARC or x86, 32-bit or 64-bit) and Linux systems (32- or 64-bit),
# provided glxspheres is installed on the target system in one of the paths below.
#$ -l arch=sol-sparc|sol-sparc64|sol-x86|sol-amd64|lx24-x86|lx24-amd64

# If VGL_DISPLAY is set by SGE, then run program with vglrun. Otherwise don't.
if [ "${VGL_DISPLAY+set}" ]; then                  If VGL_DISPLAY is set (even if null)...
     VGLRUN=/opt/VirtualGL/bin/vglrun              Then the script will use vglrun to launch application.
    if [ ! -x $VGLRUN ]; then
        VGLRUN=vglrun                              Depend on finding vglrun in the user's $PATH.
    fi
```

```
else
    VGLRUN=""
fi

if  [ -x /opt/VirtualGL/bin/glxspheres ]; then
    path=/opt/VirtualGL/bin/glxspheres
else
    echo 1>&2 "glxspheres not found on host ${HOSTNAME}"
    exit 2
fi

# Sun Grid Engine job starts vglrun which starts glxspheres
# with any arguments passed to this script.  If VGL_DISPLAY is not set,
# $VGLRUN will be the empty string, and vglrun won't be invoked.
$VGLRUN "$path" "$@"
```

Index

V

W

X

Y

Z