Sun™ Shared Visualization 1.1 Software Server Administration Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 820-3256-13
November 2008, Revision A

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

Please
Recycle

Contents

Figures

Tables

Preface

This server administration guide provides detailed information and procedures for installing the Sun™ Shared Visualization 1.1 software. This document is written for system administrators 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 provides an overview of the software that enables and enhances the Sun Shared Visualization 1.1 software.

Chapter 2 describes the hardware platforms, operating systems, and graphics accelerators that support the Sun Shared Visualization 1.1 software.

Chapter 3 discusses installing the Sun Shared Visualization 1.1 software and supporting software.

Chapter 4 provides configuration information for both Solaris and Linux based Sun Shared Visualization 1.1 servers.

Chapter 5 details information the system administrator needs to know about Advance Reservation.

Appendix A discusses constraints and behaviors of three types of Sun Ray™ network architectures and VirtualGL.

Appendix B lists predetermined configuration values for selected applications.

Appendix C provides manual procedures for some configuration steps that are handled through a script in procedures in Chapter 4.

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	#

Unless stated otherwise, syntax used in this document is `csh/tcsh`.

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
Client Administration	*Sun Shared Visualization 1.1 Software Client Administration Guide*	820-3257	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 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 Server Administration Guide, part number 820-3256-13

CHAPTER 1

Software Overview

This chapter provides an overview of the software that composes and enhances the Sun Shared Visualization 1.1 software. Topics include:

CD-ROM Contents

TABLE 1-1 describes the directory structure and contents of the Sun Shared Visualization 1.1.1 software CD-ROM.

TABLE 1-1 Directory Structure and Contents of the CD-ROM

Path From `/cdrom`	**Descriptions**
`Copyright`	U.S. English copyright notice.
`FR_Copyright`	French translation of copyright notice.
`License`	Binary Code License.
`install`	Sun Shared Visualization 1.1 software installation script.
`ThirdPartyLicenseReadme.txt`	License agreement for third-party software.
`/Docs`	Contains Sun Shared Visualization 1.1 documentation.
`/SharedVisualization_1.1`	Contains the Sun Shared Visualization 1.1 software.

TABLE 1-1 Directory Structure and Contents of the CD-ROM *(Continued)*

Path From `/cdrom`				Descriptions
	`/Solaris`			Contains client and server software for the Solaris and OpenSolaris Operating Systems.
		`install`		Sun Shared Visualization 1.1.1 software Solaris and OpenSolaris installation script.
		`remove`		Sun Shared Visualization 1.1.1 software Solaris and OpenSolaris removal script.
		`/sparc`		Contains packages and supporting patches to install the Sun Shared Visualization 1.1.1 software for Solaris SPARC® platforms.
		`/x86`		Contains packages and supporting patches to install the Sun Shared Visualization 1.1.1 software for Solaris and OpenSolaris x86 platforms.
	`/Linux`			Contains client and server software for Linux operating systems.
		`install_rpm`		Sun Shared Visualization 1.1.1 software Linux RPM installation script.
		`remove_rpm`		Sun Shared Visualization 1.1.1 software Linux RPM removal script.
		`install_deb`		Sun Shared Visualization 1.1.1 software Ubuntu (Debian) Linux installation script.
		`remove_deb`		Sun Shared Visualization 1.1.1 software Ubuntu (Debian) Linux removal script.
		`/Source`		Contains source RPMs for VirtualGL and TurboVNC.
			`/3D`	Contains a source RPM for "3D extensions to Sun Grid Engine".
			`/AR`	Contains a source RPM for Advance Reservations.
			`/VirtualGL`	Contains a source RPM for VirtualGL.
			`/vnc`	Contains a source RPM for TurboVNC.
		`/x64`		Contains directories for select Linux versions for x64 platforms.
			`/rhel-3`	Contains Shared Visualization 1.1 packages for Red Hat Linux 3.
			`/rhel-4`	Contains Shared Visualization 1.1 packages for Red Hat Linux 4.
			`/rhel-5`	Contains Shared Visualization 1.1 packages for Red Hat Linux 5.

TABLE 1-1 Directory Structure and Contents of the CD-ROM *(Continued)*

Path From `/cdrom`			Descriptions
		`/suse-9`	Contains Shared Visualization 1.1 packages for SuSE 9 and SuSE 10.
		`/ubuntu`	Contains VirtualGL, TurboJPEG, and TurboVNC packages for Ubuntu Linux.
	`/x86`		Contains directories for select Linux versions for x86 platforms.
		`/rhel-3`	Contains Shared Visualization 1.1 packages for Red Hat Linux 3.
		`/rhel-4`	Contains Shared Visualization 1.1 packages for Red Hat Linux 4.
		`/rhel-5`	Contains Shared Visualization 1.1 packages for Red Hat Linux 5.
		`/suse-9`	Contains Shared Visualization 1.1 packages for SuSE 9 and SuSE 10.
		`/ubuntu`	Contains VirtualGL, TurboJPEG, and TurboVNC packages for Ubuntu Linux.
`/Mac`			Contains client software for Mac OS X clients on Apple Macintosh x86 systems.
	`VirtualGL.dmg`		VirtualGL Mac disk image.
	`TurboVNC.dmg`		TurboVNC Mac disk image.
`/Windows`			Contains client software for the Windows XP operating system.
	`VirtualGL.exe`		Self-expanding VirtualGL installation file.
	`TurboVNC.exe`		Self-expanding TurboVNC installation file.
`/Source`			Contains source files for Advance Reservations, VirtualGL, TurboJPEG, and TurboVNC.
	`/AR`		Contains compressed files to support Advance Reservations.
	`/VirtualGL`		Contains the compressed tar file of VirtualGL.
	`/turbojpeg`		Contains the compressed tar file of TurboJPEG.
	`/vnc`		Contains the compressed tar file of TurboVNC.

Source Directory Structure of 1.1 Release

Almost all the Sun Shared Visualization Software is open source, and is available on the CD-ROM and online for free download. However, the software is not automatically installed by the product installation script. (The "3D extensions to Sun Grid Engine" are entirely scripts and configuration, so no additional source is needed for Solaris systems.)

In releases 1.1 and 1.1u1, source code is in the `Linux/Source` and `Solaris/Source` subdirectories, as shown in TABLE 1-2.

TABLE 1-2 Source Directory Structure of the 1.1 CD-ROM

Path From `/cdrom`	**Descriptions**
`/SharedVisualization_1.1`	Contains the Sun Shared Visualization 1.1 software.
`/Solaris`	Contains software for the Solaris Operating System.
`/Source`	Contains source files for Advance Reservations, VirtualGL, TurboJPEG, and TurboVNC.
`/AR`	Contains compressed files to support Advance Reservations.
`/VirtualGL`	Contains the compressed tar file of VirtualGL.
`/turbojpeg`	Contains the compressed tar file of TurboJPEG.
`/vnc`	Contains the compressed tar file of TurboVNC.
`/Linux`	Contains software for Linux operating systems.
`/Source`	Contains product source files useful on Linux OSes.
`/3D`	Contains a source RPM for "3D extensions to Sun Grid Engine".
`/AR`	Contains a source RPM for Advance Reservations.
`/VirtualGL`	Contains Source RPM and compressed tar file for VirtualGL.
`/turbojpeg`	Contains the compressed `tar` file of TurboJPEG.
`/vnc`	Contains Source RPM and compressed tar file for TurboVNC.

VirtualGL

VirtualGL is an open source software package that provides hardware-accelerated 3D rendering capabilities to thin clients. When you run a 3D application inside a thin client environment (for example, Sun Ray, VNC, Sun Secure Global Desktop, or remote X11), normally one of more of the following occurs:

- The 3D application does not work at all.
- The 3D application is forced to use a slow software 3D renderer.
- The 3D application is forced to send every 3D command and piece of 3D data over the network to be rendered on the client host.

With VirtualGL, the 3D rendering commands from the application are intercepted at run time and redirected onto the server's 3D accelerator hardware. The resulting rendered images are then read back from the 3D hardware and composited into the appropriate window on the user's desktop. This functionality produces a completely seamless shared 3D environment that performs fast enough to take the place of a dedicated 3D workstation.

VirtualGL eliminates the workstation and the network as barriers to data size. Users can now visualize gigabytes and gigabytes of data in real time without needing to copy any of the data over the network or sit in front of the machine that is rendering the data.

Modes of Operation

VirtualGL has three basic modes of operation:

- VGL Image Transport (formerly called Direct mode)
- X11 Image Transport (formerly called Raw or Proxy mode)
- Sun Ray Image Transport (formerly called Sun Ray mode)

VGL Image Transport

In VGL Image Transport (formerly called Direct mode), VirtualGL compresses the rendered output images from 3D applications directly on the 3D application server and sends the resulting compressed images (JPEG) directly to the client.

VGL Image Transport requires an X server and the VirtualGL client application (`vglclient`) to be running on the client host.

This 2D X server:

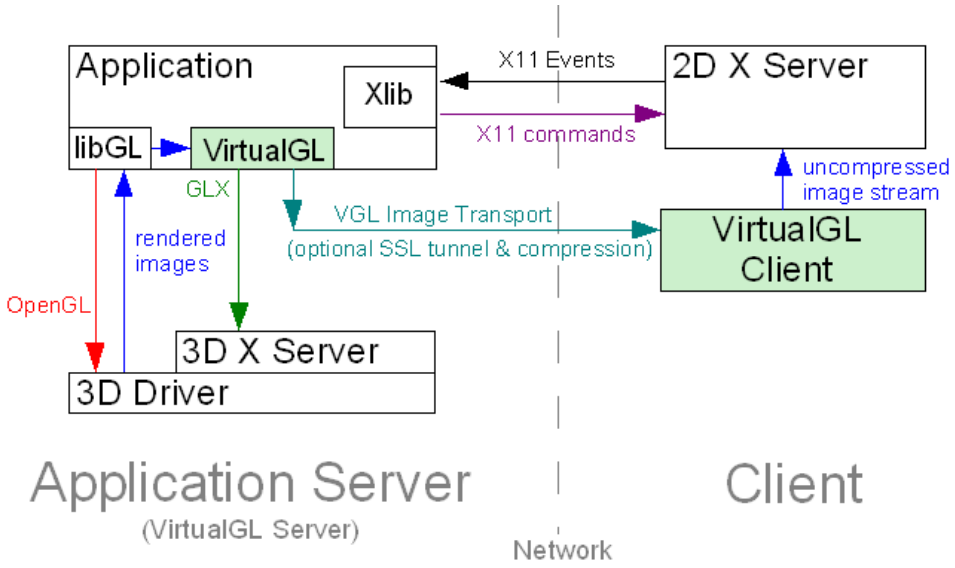
- Processes 2D drawing commands from the application
- Renders the application's user interface
- Feeds input events (key, mouse, and so on) back to the application

Meanwhile, VirtualGL:

- Intercepts the 3D commands from the application
- Reroutes the commands to the server's 3D graphics accelerator hardware
- Reads back the rendered 3D images
- Compresses the images using a high-speed image codec
- Sends the compressed images on a separate socket to the client

A separate VirtualGL client application runs on the client host. This client application decompresses the image stream from the server and composites the stream into the appropriate X window. See FIGURE 1-1.

FIGURE 1-1 VGL Image Transport



VGL Image Transport is a well-performing solution for running VirtualGL on a local area network. VGL Image Transport provides a seamless end user experience that is indistinguishable from running the application locally. VGL Image Transport is typically used to run data-intensive 3D applications in a back room and remotely interact with these applications from a laptop or a slim PC located elsewhere in the same building or facility.

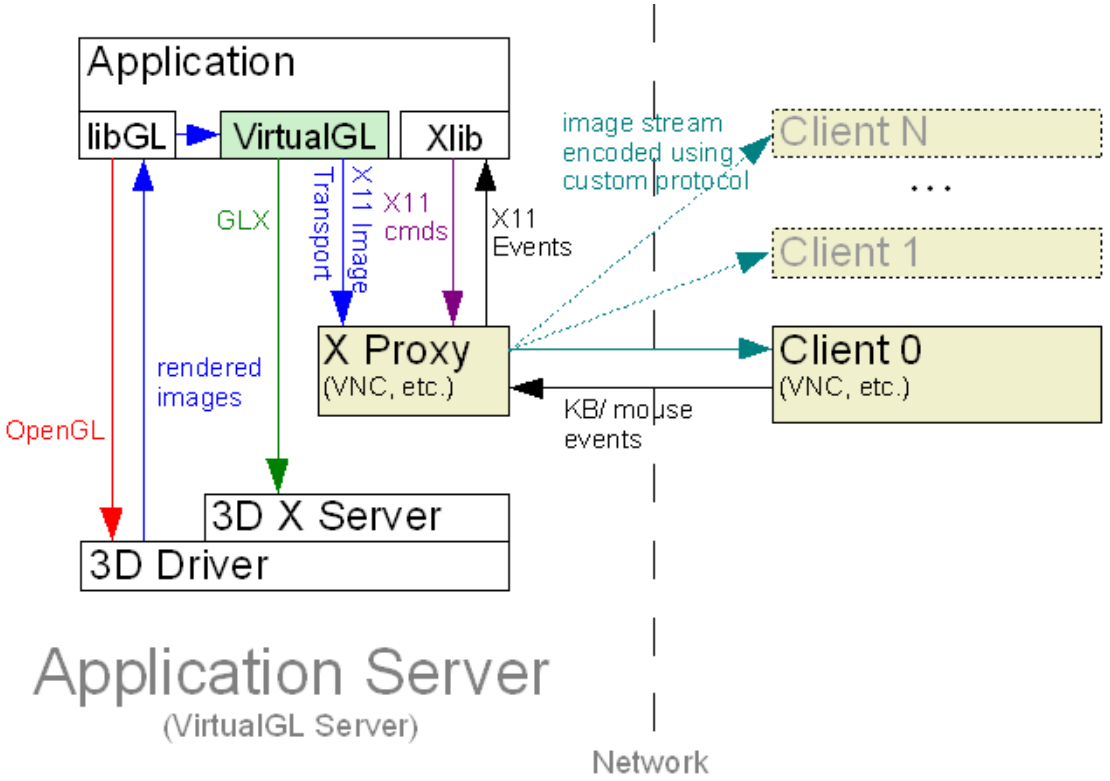
X11 Image Transport

X11 Image Transport (previously called Raw mode or Proxy mode) draws the rendered output images from 3D applications into an X proxy, such as TurboVNC, on the 3D application server. (Other X proxies may work with the open source VirtualGL but are not supported by Sun Shared Visualization Software.) The X proxy then compresses the images and sends the resulting compressed images to the client.

X11 Image Transport also can be used when the application and the 2D X server are on the same host or are connected by a high-speed, low-latency network, such as Gigabit Ethernet or faster.

With X11 Image Transport and an X proxy, the client host does not need to run an X server or `vglclient`. The 2D rendering is instead performed by an X proxy on the server host. As with VGL Image Transport, VirtualGL reroutes the 3D commands from the application to the server's 3D hardware and reads back the rendered images. But in X11 Image Transport, VirtualGL does not perform image compression. Instead, VirtualGL draws the rendered 3D images into the X proxy as uncompressed bitmaps, enabling the X proxy to compress the images and send the images to the client. See FIGURE 1-2.

FIGURE 1-2 X11 Image Transport



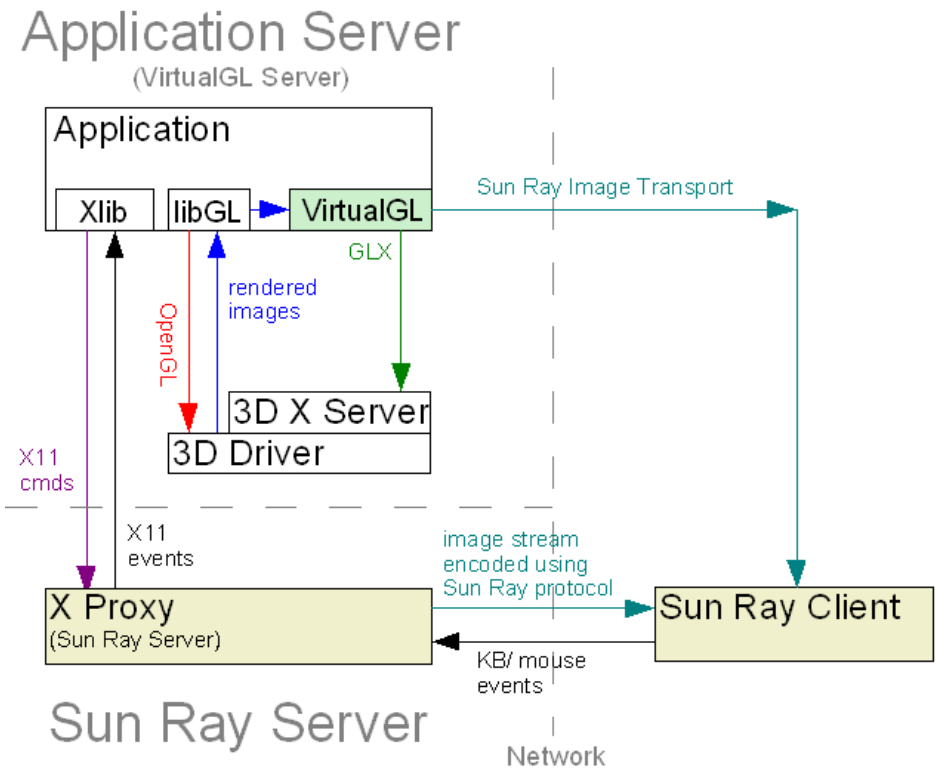
X11 Image Transport, in combination with TurboVNC, is the fastest solution for running VirtualGL on a wide area network (broadband, T1, and so on). X11 Image Transport is typically used to run data-intensive 3D applications in a back room and remotely interact with these applications from a PC located in another city.

Sun Ray Image Transport

In Sun Ray Image Transport (formerly called Sun Ray mode), VirtualGL compresses the rendered output images from 3D applications directly on the 3D application server and sends the resulting compressed images directly to the Sun Ray hardware client. A Sun Ray plug-in is installed on the 3D application server.

The Sun Ray Image Transport is a hybrid between VGL Image Transport and X11 Image Transport. The Sun Ray server acts as the X proxy, receiving and rendering 2D commands from the application. But instead of drawing the rendered 3D images into the X proxy, VirtualGL compresses these images directly using the Sun Ray image codec and sends the images directly to the Sun Ray hardware client. See FIGURE 1-3.

FIGURE 1-3 Sun Ray Image Transport



In Sun Ray environments, VirtualGL is generally not running on the same host as the Sun Ray server. Therefore, higher performance can be achieved by sending compressed images directly to the client rather than sending uncompressed images over the network through the Sun Ray server.

Sun Ray is a true thin client environment that offers significant strategic advantages over VGL Image Transport or X11 Image Transport, each of which require a UNIX or Windows XP client. The ultra-thin nature of the Sun Ray client has some disadvantages, the most notable being performance. However, VirtualGL and Sun Ray seem to be a popular solution among mechanical CAD application users, as the images generated by such applications tend to compress fairly well. Thus, acceptable performance can be achieved despite the Sun Ray client's limited image processing horsepower.

Government customers also like Sun Ray because of its security. The client has no storage devices, and thus there is no way for users to copy data off of the server unless explicitly permitted to do so.

For further information about the interactions of the Sun Ray network architecture and VirtualGL, see "Sun Ray Network Architectures and VirtualGL" on page 101.

TurboVNC

TurboVNC is a derivative of TightVNC and differs from TightVNC in the following ways:

- TurboVNC accelerates the compression and decompression of JPEG image tiles using TurboJPEG (the same JPEG codec used by VirtualGL).
- TurboVNC is tuned to provide maximum frame rates and compression ratios for 3D application workloads. In some cases, this comes at the expense of compressing 2D application workloads less efficiently than TightVNC.
- TurboVNC provides more fine-grained control over the JPEG image quality.
- TurboVNC provides (optional) double buffering on the client side to alleviate tearing artifacts in 3D and video applications.
- TurboVNC provides protocol tweaks that allow some stages of the VNC pipeline to occur in parallel. This functionality improves performance on high-latency networks.
- TurboVNC is built and tested thoroughly on Solaris platforms.

Working With VirtualGL

TurboVNC, when used with VirtualGL's X11 Image Transport, is the fastest solution for remotely displaying 3D applications across a wide-area network. TurboVNC is also an optimal solution for local-area network use. However, TurboVNC requires the user to interact with the entire remote desktop in a single client window and thus does not provide a completely seamless experience. When using the VGL Image Transport, on the other hand, each application window appears as a separate window on the client machine.

TurboVNC also supports collaboration by enabling multiple clients to connect simultaneously to a single TurboVNC server session. Users can take turns using the mouse and keyboard to control the TurboVNC session. View-only clients can observe the TurboVNC session without providing input.

Throughput

TurboVNC is capable of sending over 30 Megapixels/second over a 100 Megabit/second local area network with perceptually lossless image quality. TurboVNC can deliver 8 Megapixels/second over a 3 Megabit/second broadband connection at reduced (but usable) image quality.

Compatibility

TurboVNC is completely compatible with other VNC distributions and can be installed onto the same system as another VNC distribution without interference.

Sun Grid Engine

Sun Grid Engine (formerly called Sun N1™ Grid Engine) performs resource management and load balancing, yielding high utilization and increased project throughput. Sun Grid Engine provides a command-line interface and a graphical user interface for both users and administrators.

A Sun Grid Engine administrator can control which users or groups of users are allowed to use which execution servers at what times. An administrator also can control prioritization and scheduling policy.

Sun Grid Engine also handles starting applications on a selected execution host, so the user need not log in to the server. Job scripts can specify options to Sun Grid Engine. For example, in an environment with heterogeneous execution hosts, these options could specify which processor types and operating systems are capable of running the application.

Sun Grid Engine Graphics Additions

The Sun Shared Visualization 1.1 software extends Sun Grid Engine capabilities to allocate graphics resources. In an environment that has multiple execution hosts or multiple graphics accelerators on an execution host, Sun Grid Engine can select a suitable, lightly-loaded server to run your application. The software can also select a lightly-loaded graphics accelerator on that server.

The Sun Grid Engine administrator can configure how many jobs can run simultaneously on a server and on a graphics accelerator.

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. The requirement is to schedule compute and visualization resources at a time when the computer resources and the people to use the resources are both available. The Advance Reservation server makes this situation 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 Reservation" on page 89 and the *Sun Shared Visualization 1.1 Client Administration Guide*, 820-3257, for more information.

CHAPTER 2

Platforms

This chapter describes the hardware platforms, operating systems, and graphics accelerators that support the Sun Shared Visualization 1.1 software. Topics include:

Supported Platforms

Server Platforms

TABLE 2-1 describes the server platforms supported by the Sun Shared Visualization 1.1 software.

TABLE 2-1 Supported Server Platforms

Processor Architecture	Operating System	OS Releases
UltraSPARC®	Solaris OS	At least Solaris 8
x86	Solaris OS	At least Solaris 10 and OpenSolaris 2008.11
x86	Linux	Red Hat Enterprise Linux versions 3, 4, and 5; SuSE 9 and 10; and Ubuntu Linux 6.06 and later

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 2-2 describes the graphics accelerators supported by the Sun Shared Visualization 1.1 and 1.1.1 software for various processor architectures.

TABLE 2-2 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 and 1.1.1 software also supports Chromium graphics clusters, when the Chromium Head Node is running the Sun *Scalable* Visualization software and is configured like a 3D application server.

Client Platforms

TABLE 2-3 describes the client platforms supported by the Sun Shared Visualization 1.1 software.

TABLE 2-3 Supported Client Platforms

Processor Architecture	Minimum Supported CPU Clock Speed	Operating System	OS Releases
UltraSPARC	900MHz	Solaris OS	Solaris 8 and later
x86	1.0 GHz	Solaris OS	Solaris 10 and OpenSolaris 2008.11 and later
x86	1.0 GHz	Linux	Red Hat Enterprise Linux 3, 4, and 5; SuSE 9 and 10; and Ubuntu Linux 6.06 and later
x86	1.0 GHz	Windows	Windows XP or Vista. VGL Image Transport requires Exceed 2006 or later, or Exceed 3D for stereographic display support.
x86-based Macintosh	1.0 GHz	Mac OS X	Mac OS X 10.4 (Tiger) and 10.5 (Leopard)

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 you are using a 3D graphics accelerator, install the vendor's current OpenGL® library and drivers for that 3D accelerator.

Platform Details

This section explains the supported platforms in depth.

SPARC Platforms

These servers and clients use an UltraSPARC processor, running in either 32-bit or 64-bit mode. All SPARC platforms can use the Solaris 10, Solaris 9, or Solaris 8 Operating System. SPARC 3D application servers use the XVR-2500, XVR-1200, or XVR-600 graphics accelerators.

Appropriate software and patches for the respective graphics accelerators must be applied. TABLE 2-4 lists those patches for the graphics accelerators.

TABLE 2-4 SPARC Platform Software and Patches Respective to Graphics Accelerators

	Patches or Software for OS		
Graphics Accelerator	**Solaris 10 OS**	**Solaris 9 OS**	**Solaris 8 OS**
XVR-2500	120928	120927	N/A
XVR-1200	118708	114555	114554
XVR-600	118708	114555	114554

OpenGL Patches for Solaris SPARC

In order to ensure correct operation of Sun Shared Visualization 1.1 and 1.1.1, the version of Sun OpenGL installed on a SPARC 3D application server might need to be patched. TABLE 2-5 lists the minimum OpenGL patch levels that should be used with Sun Shared Visualization 1.1 or 1.1.1.

TABLE 2-5 Patches for Versions of Solaris SPARC OpenGL

OpenGL Version	OpenGL Patches
OpenGL 1.5 (recommended)	120812
OpenGL 1.3 (64-bit)	113887
OpenGL 1.3 (32-bit)	113886

The most recent *Sun Shared Visualization 1.1.1 Software Release Notes* identify the required revision levels of required patches. All patches can be downloaded from:

`http://sunsolve.sun.com/patches`

GLP Access on Solaris SPARC Servers

A Solaris SPARC 3D application server with OpenGL 1.5 and an XVR-2500, XVR-1200, or XVR-600 graphics accelerator can be configured to use those devices through GLP without having to start a 3D X server on the graphics accelerators.

The Sun OpenGL library for Solaris SPARC systems has a special extension called GLP, which allows VirtualGL to directly access a 3D graphics card even if there is no X server running on the card. GLP greatly improves the overall security of the VirtualGL server by eliminating the need to grant X server access to VirtualGL users. In addition, GLP makes it easy to assign VirtualGL jobs to any graphics card in a multicard system.

When using GLP, the VirtualGL architecture changes in these ways:

- VGL Image Transport with GLP is as shown in FIGURE 2-1 (a change from FIGURE 1-1 on page 6)
- X11 Image Transport with GLP is as shown in FIGURE 2-2 (a change from FIGURE 1-2 on page 8)
- GLP can also be used with the Sun Ray Image Transport in place of GLX (shown in FIGURE 1-3 on page 9).

GLP access can be enabled by running `vglserver_config` on a Solaris SPARC 3D application server. For more information, see "To Configure a Solaris SPARC Server to Use VirtualGL With GLP and Without a 3D X Server" on page 57.

FIGURE 2-1 VGL Image Transport With GLP Access to Graphics Accelerator

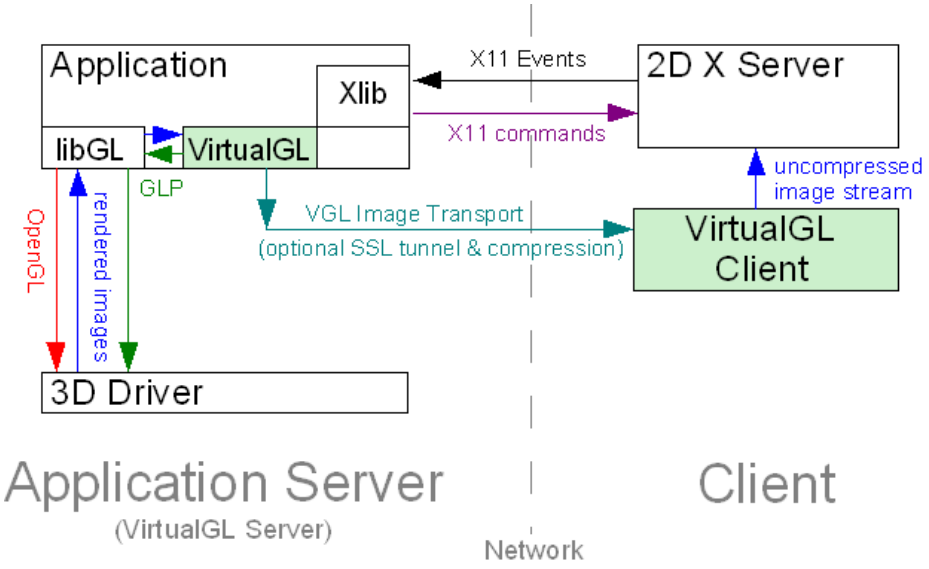
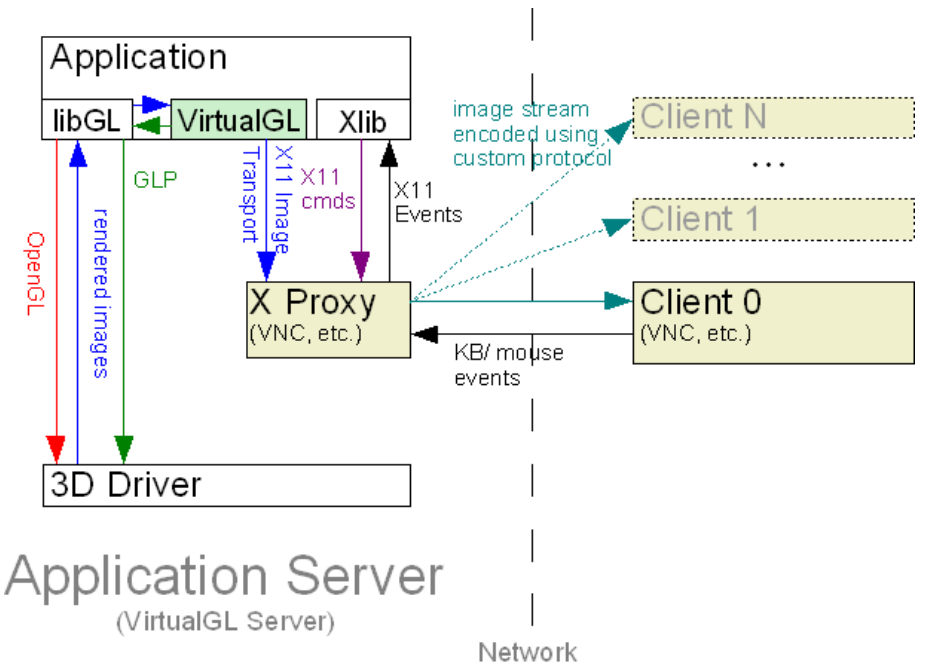


FIGURE 2-2 X11 Image Transport With GLP Access to Graphics Accelerator



x64 Platforms

These servers and clients use the AMD Opteron processor or an Intel processor that implements the Intel 64 instruction set. These processors run in 64-bit mode and are supported by the Solaris 10 and OpenSolaris Operating Systems. (The Solaris 9 OS and Solaris 8 OS do not support x64 processors.) Additionally, Red Hat Linux versions 5, 4, and 3, SuSE Linux 10 and 9, and Ubuntu Linux 6.06 and later support 64-bit instructions on x64 processors.

The x64 server systems are supported with NVidia Quadro and Quadro Plex graphics accelerators. Software and patches for these graphics accelerators are available at:

http://sunsolve.sun.com

http://www.nvidia.com

x86 Platforms

These servers and clients use the AMD Athlon or Intel x86 processor. These processors run in 32-bit mode and are supported by the OpenSolaris 2008.11 and Solaris 10, Solaris 9, and Solaris 8 Operating Systems. Additionally, Red Hat Linux versions 5, 4, and 3; SuSE Linux 10 and 9; and Ubuntu Linux 6.06 and later support these processors.

Sun Shared Visualization 1.1 software supports x86 server systems configured with NVidia Quadro or Quadro Plex graphics accelerators. Software and patches for these graphics accelerators are available at:

http://sunsolve.sun.com

http://www.nvidia.com

x64 and x86 Clients

The information in this section applies to x64 and x86 client platforms.

x64 and x86 clients can use most graphics accelerators that support:

- 24- or 32-bit pixel true color display
- 3D graphics acceleration, for stereographic display support

UNIX graphics applications require an X server, but the Windows operating system does not ship with an X server. If you are using the VGL Image Transport with a Windows XP or Vista client, then the client must have the 32-bit version of the Exceed 2006 X server (or a later version) installed. If you want to remotely display a

3D application using quad-buffered stereographic rendering or transparent overlays, then 32-bit Exceed 3D 2006 (or later) must be installed on the Windows client. Exceed 3D is not required in order to use anaglyphic stereographic rendering.

Note – TurboVNC can be used on a Windows client without Exceed.

Use `nvidia` Rather Than `nv` Driver

Note – Use the latest `nvidia` driver for NVidia graphics accelerators. Do not use the `nv` driver that might have come with your operating system. Running `nvidia-xconfig -a` is a common operating system technique to enable the `nvidia` driver for all NVidia GPUs.

Sun Ray Platforms

The Sun Ray client is stateless and unconfigurable. The client depends on the Sun Ray server for display information. In the Sun Shared Visualization 1.1 environment, the Sun Ray client can receive display information from VirtualGL, as well as from its Sun Ray server. The Sun Ray server need not have a graphics accelerator installed.

CHAPTER 3

Installing the Software

This chapter discusses installing the Sun Shared Visualization 1.1 software and supporting software. Topics include:

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

Installation Summary

The Sun Shared Visualization Software 1.1 supports different use models and administration models, as described in "Software Overview" on page 1. The supplied installation script installs the software needed by most sites and optionally installs software for use with Sun Grid Engine.

Note – If you are sure you do not need to install Sun Grid Engine and Sun Shared Visualization optional extensions for Sun Grid Engine, you can begin installation with the procedures in "Installing Sun Shared Visualization 1.1 Software" on page 42.

Planning Your Sun Shared Visualization Environment

Before installation, identify your shared visualization resources:

- Graphics application servers, which run applications under control of VirtualGL and optionally run TurboVNC server sessions on demand.
- Shared graphics accelerator devices, which are installed on the graphics application servers.
- Shared visualization client hosts.
- Shared visualization client users, who start VirtualGL and TurboVNC clients and optionally submit jobs to Sun Grid Engine.

Sun Grid Engine

Note – Sun Grid Engine was formerly called Sun N1 Grid Engine. Some documentation for the current product includes the earlier name.

Sun Shared Visualization 1.1 software includes optional software for use with Sun Grid Engine:

- Sun Grid Engine graphics extensions
- Advance Reservation facility for Sun Grid Engine

You might not need Sun Grid Engine if your site has only one Sun Shared Visualization 1.1 server and that server has only one graphics accelerator. However, if you have multiple servers or multiple graphics accelerators, Sun Grid Engine can allocate these resources to users with load balancing.

If you are using Sun Grid Engine, you need to determine which hosts are:

- Queue master server for Sun Grid Engine
- NFS server for the Sun Grid Engine installation

All Sun Grid Engine hosts will NFS mount the SGE installation from this NFS server.

- Sun Grid Engine execution hosts

  All graphics servers should be execution hosts, but you might have additional execution hosts that do not offer shared visualization services.

- Sun Grid Engine administration hosts
- Sun Grid Engine submit hosts

  All shared visualization clients are typically Sun Grid Engine submit hosts.

- Advance Reservation server

  If the Advance Reservation facility is installed, only one host must be the Advance Reservation server.

Note – A single host can have multiple roles.

Note – Sun Shared Visualization 1.1 software does not include Sun Grid Engine. This software is available from:
`http://www.sun.com`

Summary of Preparatory Steps

1. Install and configure hardware, operating systems, and windowing systems on each host.

   Ensure that the graphics servers have supported hardware and operating system versions. Ensure that the latest drivers for the graphics accelerators are installed and patched.

2. Install visualization applications on graphics servers and computation applications on Sun Grid Engine execution hosts.

   Or, applications can be remotely mounted as needed.

Note – Licensing and appropriate use of all visualization applications is entirely the responsibility of the user.

---

3. If you are using Sun Ray thin clients, install the Sun Ray Server software and configure any Sun Ray servers.

   Refer to Appendix A for additional guidelines.

4. If your site is using Sun Grid Engine, install Sun Grid Engine on the site's NFS server.

   Instructions to help you install and configure Sun Grid Engine are provided in "Installing the Sun Grid Engine Software" on page 24.

Summary of the Installation Process

1. If your site is using Sun Grid Engine, install the Sun Shared Visualization 1.1 software, including the optional software, on the NFS server host for the Sun Grid Engine installation.

   See "Installing Sun Shared Visualization 1.1 Software" on page 42.

2. Install the Sun Shared Visualization 1.1 software on all graphics servers.

   The Sun Grid Engine optional software is not installed on these servers. Instead, these servers mount the optional software from the NFS server.

3. Configure each system planned to be a graphics server.

   See Chapter 4.

4. Test use of VirtualGL (and, optionally, TurboVNC) on each Sun Shared Visualization 1.1 server.

   See Chapter 3 of the *Sun Shared Visualization 1.1 Software Client Administration Guide*.

5. If your site is using Sun Grid Engine, add graphics to Sun Grid Engine.

   See "Adding Graphics to Sun Grid Engine" on page 64.

6. If your site is using Sun Grid Engine and Advance Reservation, configure the Advance Reservation server on a single host.

   See "Advance Reservation" on page 89.

Installing the Sun Grid Engine Software

This section describes installing the Sun Grid Engine software. These instructions are streamlined for installations particular to the Sun Shared Visualization 1.1 software.

Complete Sun Grid Engine documentation, including an installation guide, is available at:

`http://docs.sun.com/app/docs/coll/1017.3`

▼ To Prepare to Install the Sun Grid Engine Software

This procedure is for installations on all Solaris and Linux servers.

1. **Determine which host is to be the queue master (`qmaster`) and which host is to be the NFS server for your grid.**

   If the resources are available, the same host can perform both roles.

2. **Determine which hosts are to be the execution hosts for your grid.**

   If the resources are available and these systems are configured with graphics accelerators, the execution hosts can also be the graphics servers.

   Note – Execution hosts need the korn shell, `ksh`. Solaris hosts include `ksh` by default, but Linux hosts might need `ksh` to be installed.

   ---

3. **Determine your installation `SGE_ROOT` directory.**

   The package default is `/gridware/sge`, however the Sun Grid Engine documentation calls this `<sge_root>` or `/sge_root`. These instructions use the variable, `$SGE_ROOT`.

4. **Become superuser of the NFS server and declare the variable:**

   ```
   # setenv SGE_ROOT /gridware/sge
   ```

   If you chose a different installation `root` directory in Step 3, type that directory name instead of `/gridware/sge`.

5. **Create the base directory for `$SGE_ROOT` if the path has multiple directory components:**

   ```
   # mkdir /gridware
   ```

6. **Determine an SGE administrative login that can be used on all systems intended to be administration hosts.**

For example, you might plan to use these parameters:

Parameter	Value
Name	`sgeadmin`
Group	`adm (4)`
Home directory	`$SGE_ROOT` or `/gridware/sge` (if that is your `SGE_ROOT` choice)
User ID	`530`

The Sun Grid Engine administrator can have a different user ID than `sgeadmin`. However, the administrative user ID (`530` in this example) must be available across all hosts in the grid.

On SuSE hosts, group 4 (`adm`) might not already be defined in `/etc/group`. In that case, you need to add that group.

7. **Create the `sgeadmin` user on the NFS server for your grid.**

   Use the values you selected in Step 6, as in this example::

```
# useradd -u 530 -g 4 -d $SGE_ROOT -m -s /bin/tcsh -c "Sun Grid Engine Admin" sgeadmin
```

8. **Assign the `sgeadmin` user a password:**

```
# passwd sgeadmin
```

9. **Append the following lines to the `sgeadmin .cshrc` file:**

```
if ( $?prompt == 1 ) then
       if ( -f /gridware/sge/default/common/settings.csh ) then
            source /gridware/sge/default/common/settings.csh
       endif
endif
```

Replace `/gridware/sge` with the value of `$SGE_ROOT` if different.

Note – You cannot use the `$SGE_ROOT` variable in Step 9, as the variable will not be set in a fresh shell until the `settings.csh` file is sourced.

You might choose to do the same for root's `.cshrc` or `.tcshrc`, or the equivalent file for root's shell.

10. **Continue the installation of software on the NFS server by performing one of these procedures:**
    - "To Install the Software on a Solaris or OpenSolaris System" on page 27
    - "To Install the Software on a Linux System" on page 31

▼ To Install the Software on a Solaris or OpenSolaris System

1. **Permit `$SGE_ROOT` to be shared (exported) by the NFS server.**

   If your base directory of `$SGE_ROOT` is already shared, you do not need to perform this step.

   On the Solaris NFS server, append the following line to the `/etc/dfs/dfstab` file:

   ```
   share -F nfs /gridware
   ```

   where `/gridware` is the base directory of your `$SGE_ROOT`.

2. **Inform the operating system of the changes you have made:**
   - For an OpenSolaris or Solaris 10 OS:

     ```
     # svcadm -v restart nfs/server
     ```

   - For Solaris releases earlier than Solaris 10:

     ```
     # /etc/init.d/nfs*server stop ; /etc/init.d/nfs*server start
     ```

3. **If the system automounts using the `hosts` map, you can test the accessibility of the `$SGE_ROOT` directory from other systems on the network with this command:**

   ```
   # ls /net/nfsserverhostname/$SGE_ROOT
   ```

4. **From each server in the grid, access the NFS server's `$SGE_ROOT` as each server's `$SGE_ROOT` using `/etc/vfstab`, `/etc/fstab`, or automounting.**

Note – Submit hosts (client machines) also need to mount the NFS server's `$SGE_ROOT`.

Execution hosts must not mount the NFS server with the `nosuid` option, as `setuid` is needed by Sun Grid Engine's `rlogin` and `rsh` for its `qrsh` command to work properly.

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`. If the NFS client prefers NFS version 4 but the NFS server does not, this line might also need the `nfs=3` option.

b. Ensure that the `/etc/auto_master` or the `auto_master` NIS map contains this entry:

```
/-     auto_direct
```

c. Restart the automounter:

- For an OpenSolaris or Solaris 10 OS:

```
# svcadm -v restart autofs
```

- For Solaris releases earlier than Solaris 10:

```
# /etc/init.d/autofs stop ; /etc/init.d/autofs start
```

Note – Use network automounting only if doing so also mounts `suid`. Sun Grid Engine requires certain components to be `set-uid` in order for `qrsh` to work properly.

On a system that automounts using the `hosts` map, 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
```

5. **Determine port numbers.**

   You must determine an available port on the `qmaster` system. Sun Grid Engine components will use this port to communicate with the `qmaster` daemon. This port must be a single port number that is available on all current or prospective submit and execution hosts in your grid.

   These port numbers can be any value, but the following port numbers have been assigned by the Internet Assigned Number Authority (IANA):

Name	Number
`sge_qmaster`	6444/tcp
`sge_execd`	6445/tcp

   Note – For more information about IANA, see:
   `http://www.iana.org/assignments/port-numbers`

   If you are running a firewall on any execution host, ensure that the execution daemon's port allows traffic in.

6. **Communicate the port numbers to the hosts.**

   These port numbers can be communicated to the hosts involved either by inserting the port numbers into every host's `/etc/inet/services` or `/etc/services` file, or by setting Sun Grid Engine environment variables. The latter method, detailed in Step 4 of "To Complete the Software Installation" on page 34, is more convenient, because each Sun Grid Engine user already needs to use a Sun Grid Engine environment setup file. If you allow Sun Grid Engine to use this setup file, you will not have to add `sge` entries into every host's services file.

   To use this environment variable technique, set these environment variables before you invoke `./install_qmaster` in Step 2 of "To Complete the Software Installation" on page 34. Use the port numbers determined in Step 5 in place of `6444` and `6445` in these commands:

```
# setenv SGE_QMASTER_PORT 6444
# setenv SGE_EXECD_PORT   6445
```

   The lines you include in the setup file for Sun Grid Engine will be executed by Step 5 of "To Complete the Software Installation" on page 34. (After installation, you will need to ensure that the setup file's set and export environment variables are naming `SGE_QMASTER_PORT` and `SGE_EXECD_PORT`.)

7. **As superuser of the NFS server, install the Sun Grid Engine packages into** `$SGE_ROOT`.

The NGS server will need both Sun Grid Engine architecture-independent common files and architecture-dependent files for the architecture of every submit and execution host. (Each architecture is a pairing of processor instruction set and operating system.) You might also choose to install documentation files.

These files can be installed from Solaris packages on a Solaris system or from RPM packages on a Linux system. Files for additional nonnative architectures need to be installed from `tar` bundles, which is explained in Step 1 of "To Complete the Software Installation" on page 34.

Refer to TABLE 3-1, which lists commonly used Sun Grid Engine 6.1 Solaris software packages and the download files that contain those packages. If you are installing a release other than Sun Grid Engine 6.1, the download file names will refer to that version instead of reading `6_1`. Also, newer versions of Sun Grid Engine might use file names that say `sge` instead of `n1ge`.

TABLE 3-1 Sun Grid Engine 6.1 Solaris Software Packages

Application	Download File	Package Name	Description
Common	`nlge-6_1-common.zip`	`SUNWsgeec`	Sun Grid Engine architecture-independent common files
	`nlge-6_1-common.zip`	`SUNWsgeed`	Sun Grid Engine documentation files (some SGE releases do not include this package, because documentation is provided online)
SPARC	`nlge-6_1-bin-solaris-sparcv9.zip`	`SUNWsgeex`	Solaris 64-bit binaries for SPARC
X86	`nlge-6_1-solaris-i586.zip`	`SUNWsgeei`	Solaris 32-bit binaries for x86
X64	`nlge-6_1-bin-solaris-x64.zip`	`SUNWsgeeax`	Solaris 64-bit binaries for x64
Common but Optional	`nlge-6_1-arco.zip`	`SUNWsgeea`	Accounting and Reporting Console (ARCo) for all architectures (optional)

To install Sun Grid Engine from the packages you selected, first unzip the download files. then install each package by typing a `pkgadd` command line such as this:

```
# pkgadd -d downloaddirectory  packagename
```

For all packages, answer all questions about where Sun Grid Engine should be installed with the value you chose for `$SGE_ROOT`.

Note – Some of these packages install `setuid` or `setgid` files for which `pkgadd` asks for permission. This permission should be granted.

8. **Perform the steps in** "To Complete the Software Installation" on page 34**.**

▼ To Install the Software on a Linux System

1. **Permit** `$SGE_ROOT` **to be shared (exported) by the NFS server.**

   If your base directory of `$SGE_ROOT` is already shared, you do not need to perform this step.

   On the Linux NFS server, append the following line to the `/etc/exports` file:

```
/gridware      *(rw,sync,no_root_squash)
```

   where `/gridware` is the base directory of your `$SGE_ROOT`.

2. **Inform the operating system of the changes you have made:**

   - For SuSE Linux:

```
# /etc/init.d/nfs*server stop ; /etc/init.d/nfs*server start
```

   - For Red Hat Linux:

```
# /etc/init.d/nfs restart
```

3. **If the system automounts using the** `hosts` **map, you can test the accessibility of the** `$SGE_ROOT` **directory from other systems on the network with this command:**

```
# ls /net/nfsserverhostname/$SGE_ROOT
```

4. **From each server in the grid, access the NFS server's** `$SGE_ROOT` **as each server's** `$SGE_ROOT` **using** `/etc/vfstab`**,** `/etc/fstab`**, or automounting.**

Note – Submit hosts (client machines) also need to mount the NFS server's `$SGE_ROOT`.

Execution hosts must not mount the NFS server with the `nosuid` option, as `setuid` is needed by Sun Grid Engine's `rlogin` and `rsh` for its `qrsh` command to work properly.

a. Add the following line to the `/etc/fstab` file:

```
nfsserverhostname:/gridware    /gridware  nfs        auto,suid,bg,intr          0 0
```

Your Linux system might also need the `no_root_squash` option in this line.

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` is in the `hosts` entry in the `/etc/nsswitch.conf` file. Mounting occurs before the NIS name service is started. The first hostname on the `/etc/hosts` line for the execution host itself should not include a domain.

5. Determine port numbers.

You must determine an available port on the `qmaster` system. Sun Grid Engine components will use this port to communicate with the `qmaster` daemon. This port must be a single port number that is available on all current or prospect submit and execution hosts in your grid.

These port numbers can be any value, but the following port numbers have been assigned by the Internet Assigned Number Authority (IANA):

Name	Number
`sge_qmaster`	6444/tcp
`sge_execd`	6445/tcp

Note – For more information about IANA, see:
http://www.iana.org/assignments/port-numbers

If you are running a firewall on any execution host, ensure that the execution daemon's port allows traffic in.

6. **Communicate the port numbers to the hosts.**

   These port numbers can be communicated to the hosts involved either by inserting the port numbers into every host's `/etc/inet/services` or `/etc/services` file or by setting Sun Grid Engine environment variables. The latter method, detailed in Step 4 of "To Complete the Software Installation" on page 34, is more convenient, because each Sun Grid Engine user already needs to use a Sun Grid Engine environment setup file. If you allow Sun Grid Engine to use this setup file, you will not have to add `sge` entries into every host's services file.

   To use this environment variable technique, set these environment variables before you invoke `./install_qmaster` in Step 2 of "To Complete the Software Installation" on page 34. Use the port numbers determined in Step 5 in place of `6444` and `6445` in these commands:

```
# setenv SGE_QMASTER_PORT 6444
# setenv SGE_EXECD_PORT   6445
```

   The lines you include in the setup file for Sun Grid Engine will be executed by Step 5 of "To Complete the Software Installation" on page 34. (After installation, you will need to ensure that the setup file's set and export environment variables are naming `SGE_QMASTER_PORT` and `SGE_EXECD_PORT`.)

7. **As superuser of the NFS server, install the Sun Grid Engine packages into `$SGE_ROOT`.**

   The NGS server will need both Sun Grid Engine architecture-independent common files and architecture-dependent files for the architecture of every submit and execution host. (Each architecture is a pairing of processor instruction set and operating system.) You might also choose to install documentation files.

   These files can be installed from RPM packages on a Linux system. Files for additional nonnative architectures need to be installed from `tar` bundles, which is explained in Step 1 in "To Complete the Software Installation" on page 34.

Refer to TABLE 3-2, which lists commonly used Sun Grid Engine 6.1 Linux software RPM packages and the download files that contain those packages. If you are installing a release other than Sun Grid Engine 6.1, the download file names will refer to that version instead of reading `6_1`. Also, newer versions of Sun Grid Engine might use file names that say `sge` instead of `n1ge`.

TABLE 3-2 Sun Grid Engine 6.1 Linux Software RPM Packages

Application	RPM Package	Description
Common	`sun-n1ge-common-6.1.0.noarch.rpm`	Sun Grid Engine architecture-independent common files, including documentation files
X64	`sun-n1ge-bin-linux24-x64-6.1.0.x86_64.rpm`	Linux kernel 2.4 or 2.6, glibc >= 2.3.2, for AMD Opteron or Intel EM64T
X86	`sun-n1ge-bin-linux24-i586-6.1.0.i386.rpm`	Linux kernel 2.4 or 2.6, glibc >= 2.3.2, for 32-bit x86
Common but Optional	`sun-n1ge-arco-6.1.0.noarch.rpm`	Accounting and Reporting Console (ARCo) for all architectures, not needed for the core product (optional).

To install each of the RPM packages you selected, type an `rpm` command line such as this:

```
# rpm -iv /path-to-rpm-file/sun-n1ge-rest-of-filename.rpm
```

8. **Perform the steps in** "To Complete the Software Installation" on page 34**.**

▼ To Complete the Software Installation

This procedure is for installations on all Solaris and Linux servers.

1. **Install additional Sun Grid Engine `tar` bundles of files needed by hosts with a different operating system than the NFS server.**

TABLE 3-3 lists Sun Grid Engine 6.1 software `tar` bundles, which can install nonnative software on an OpenSolaris, Solaris, or Linux NFS server. Use these bundles to install software on an NFS server as needed to support hosts with a different operating system. (Newer versions of Sun Grid Engine might use file names that say `sge` instead of `n1ge`.)

TABLE 3-3 Sun Grid Engine 6.1 Software `tar` Bundles

Name of `tar` File Bundle	Description
`nlge-common.tar.gz`	Architecture independent files (required, but was already installed from packages on the NFS server)
`nlge-6_1-bin-linux24-amd64.tar.gz`	Linux kernel 2.4 or 2.6, `glibc` >= 2.3.2, for AMD Opteron and Intel EM64T
`nlge-6_1-bin-linux24-i586.tar.gz`	Linux kernel 2.4 or 2.6, `glibc` >= 2.2.5, for 32-bit x86
`nlge-6_1-bin-solaris-sparcv9.tar.gz`	Solaris 8 and higher, for 64-bit SPARC
`nlge-6_1-bin-solaris-i586.tar.gz`	Solaris 9 and higher, for 32-bit x86
`nlge-6_1-bin-solaris-x64.tar.gz`	Solaris 10, for 64-bit x64 (such as AMD Opteron)
`nlge-6_1-bin-windows-x86.tar.gz`	Microsoft Windows*
`nlge-6_1-arco.tar.gz`	Accounting and Reporting Console (ARCo) for all architectures, not needed for the core product
`swc_linux_2.2.5.tar.gz`	Sun Web Console, required for ARCo, Linux, for 32-bit x86
`swc_solx86_2.2.5.tar.gz`	Sun Web Console, required for ARCo, Solaris, for x86
`swc_sparc_2.2.5.tar.gz`	Sun Web Console, required for ARCo, Solaris, for 64-bit SPARC

* When NFS mounts onto a Microsoft Windows client, `qrsh` will not work. A combination of locally installing on the Windows client and copying configuration from the `qmaster` host to the Windows client might enable `qrsh` to work on that client. Seek Sun Grid Engine support if this is necessary.

After you download the additional software you need, you can install the contents of each `tar.gz` file in the $SGE_ROOT directory with a command such as this:

```
# gunzip -c nlge-6_1-platform.tar.gz | (cd $SGE_ROOT; tar xf -)
```

If you installed any of the tar bundles mentioned in this step, you will need to answer **n** when the installation script asks (as in Step 3):

```
Did you install this version with >pkgadd< or did you already
verify and set the file permissions of your distribution (enter: y)
```

2. **On the queue master host, type:**

```
# cd $SGE_ROOT ; ./install_qmaster
```

The Sun Grid Engine installation script begins.

3. **The script prompts you for information and requests confirmation of selected values.**

As you progress through the script, consider the following:

- The *Sun N1 Grid Engine 6 Installation Guide* has a table to help plan and record the answers to the questions asked during installation. For the simplest installation, accept all the defaults not discussed in the following text, unless your `$SGE_ROOT` is not `/gridware/sge`.
- The installation script asks: "`Do you want to install Grid Engine as admin user >sgeadmin<? (y/n)`". Answer **y**, so that all spool files are created as owned by that user. This answer avoids a problem where an execution host's `root` becomes `nobody` over NFS and therefore cannot access the spooling directories.

Note – The installation script might instead ask this question: "`Do you want to install Grid Engine under a user id other than >root<? (y/n) [y]`". Answer **y**. Later, you are asked for the user ID, which can be `sgeadmin` (as created in Step 6 of "To Prepare to Install the Sun Grid Engine Software" on page 25).

- The installation script asks: "`Did you install this version with >pkgadd< or did you already verify and set the file permissions of your distribution (enter: y)`". If you installed exclusively from packages, answer **y**. If you installed even partially from `tar` files (as in Step 1) or other means, answer **n**, and the `install_qmaster` script sets the file permissions appropriately.
- The installation script asks: "`Are all hosts of your cluster in a single DNS domain (y/n)`". Unless you are certain that you need domain checking, answer **y**. Sun Grid Engine then ignores domain components when comparing hostnames.

Execution hosts and the queue master must agree on the primary name of the execution host. If the execution host and the queue master do not agree on hostnames, a `host_aliases` file in the `$SGE_ROOT/default/common` directory (substitute your cell's name for `default`) enables SGE to understand that certain names are equivalent. For example, a `host_aliases` file might include this line:

```
myhost1 my1 myhost1-ib my1-ib
```

Every host name on this line is considered equivalent to the first name on the line (`myhost1`), which is the primary host name. After changing the `host_aliases` file, restart SGE daemons (`sge_qmaster` and `sge_execd` on all execution hosts). For more details, see the Sun Grid Engine man page for `host_aliases` (5).

In addition, Sun Grid Engine requires that a host's unique hostname is associated with a true IP address, not the localhost address 127.0.0.1.

- Select to use the BerkeleyDB, but do not configure a separate BerkeleyDB server.
- If your site uses NIS, a usable group ID range can be determined by studying the output of:

```
# ypcat -k group.bygid | sort -n | more
```

Or, ask your administrator for a reasonable range of unused group IDs. Sun Grid Engine uses the group IDs for each of the parallel jobs that are running at a given time.

- When prompted for administrative and submit hosts, include the name of the queue master host as an administrative and submit host, unless you forbid submissions from that host.
- You can create a shadow host that takes over for the `qmaster` if it becomes unavailable. This action is optional.
- Use the following command to add administrative hosts (which might be configured to be execution hosts) if those hosts were omitted:

```
# qconf -ah hostname, anotherhost
```

- You can display the administrative host list by typing:

```
# qconf -sh
```

- You can add submit hosts by typing:

```
# qconf -as myhost,anotherhost,stillmore
```

- Typing the following displays the submit host list:

```
# qconf -ss
```

4. **Update environment variables in settings files.**

   If you decided to communicate the port numbers to all SGE hosts using SGE's environment setup file, you now need to assure that SGE sets the correct port numbers for environment variables `SGE_QMASTER_PORT` and `SGE_EXECD_PORT`. (You would have made that choice at Step 6 of "To Install the Software on a Solaris or OpenSolaris System" on page 27 or Step 6 of "To Install the Software on a Linux System" on page 31, and would have determined the port numbers in the step before these steps.)

   You might find that the proper variable values were written when you ran `install_qmaster`.

   a. **Edit the SGE settings file for `csh` or `tcsh`.**

      The file is `$SGE_ROOT/default/common/settings.csh`.

   b. **In the `settings.csh` file, look for lines such as these:**

```
unsetenv SGE_QMASTER_PORT
unsetenv SGE_EXECD_PORT
```

      If you find such lines, change them to use your port numbers.

      You determined the port numbers in Step 5 of "To Install the Software on a Solaris or OpenSolaris System" on page 27 or Step 5 of "To Install the Software on a Linux System" on page 31. For example, change the lines to the following:

```
setenv SGE_QMASTER_PORT 6444
setenv SGE_EXECD_PORT   6445
```

   c. **Edit the SGE settings file for `sh`, `bash`, and `ksh`.**

      The file is `$SGE_ROOT/default/common/settings.sh`

d. In the `settings.sh` file, look for lines such as these:

```
unset SGE_QMASTER_PORT
unset SGE_EXECD_PORT
```

If you find such lines, change them to use your port numbers.

For example, change the lines to the following:

```
SGE_QMASTER_PORT=6444;    export SGE_QMASTER_PORT
SGE_EXECD_PORT=6445;      export SGE_EXECD_PORT
```

The settings files contain the lines to unset these environment variables by default. This default behavior is desirable if you if had instead decided to enter the port numbers in every SGE host's `/etc/services` or `/etc/inet/services` file.

5. Source the file to set up your environment to use Sun Grid Engine.

- For `tcsh/csh` users, type:

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

Substitute `/gridware/sge` with your value of `$SGE_ROOT`. Consider having `root`'s `.login` do so.

- For `sh/bash/ksh` users, type:

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

Substitute `/gridware/sge` with the `$SGE_ROOT`. Consider having `root`'s `.profile` or `.bashrc` do so.

6. Create the `sgeadmin` user on each of the other administration hosts of the grid:

```
# useradd -u 530 -g 4 -d $SGE_ROOT -s /bin/tcsh -c "Sun Grid Engine Admin" sgeadmin
```

Note – Unlike Step 7 of "To Prepare to Install the Sun Grid Engine Software" on page 25, the `-m` option is not needed for these other administration hosts. Assign the `sgeadmin` a password, as in Step 8 of that procedure.

Alternatively, you can add the `sgeadmin` entries to the respective `/etc/passwd` and `/etc/shadow` files.

7. **As superuser on every execution host, set the `SGE_ROOT` environment variable and then type:**

```
# cd $SGE_ROOT ; ./install_execd
```

You might need to create the execution host's default spooling directory. As superuser on the NFS server, type:

```
# mkdir $SGE_ROOT/default/spool/exec-hostname
```

The same value for *exec-hostname* is needed in the procedure "To Set Up Sun Grid Engine Environment Variables" on page 41

8. **After the environment is set up, submit a test job.**

To specify the job to execute on your host:

```
exechost% qsub -q all.q@`hostname` $SGE_ROOT/examples/jobs/simple.sh
exechost% qstat -f
```

Job output and errors are in the initiating user's home directory, with filenames similar to the following:

```
simple.sh.e1  simple.sh.o1
```

Note – If you run the job as `root`, these files are in the execution host's root directory. If you do not know which host executed the job, you do not know which root directory the files are in. Therefore, submit jobs as a user whose home directory is in one place irrespective of execution host or specify the execution hostname explicitly.

▼ To Set Up Sun Grid Engine Environment Variables

- **Use one of the following commands:**
  - For `tcsh` and `csh` users, type:

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

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

  - For `sh`, `bash`, and `ksh` users, type:

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

    Substitute `/gridware/sge` with your `$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 `$SGE_CELL` to `$COMMD_PORT`.

Messages from Sun Grid Engine can be found in:

- `/tmp/qmaster_messages` (during Sun Grid Engine queue master startup)
- `/tmp/execd_messages` (during Sun Grid Engine exec daemon startup)

After the startup the daemons log messages in the spool directories.

- Sun Grid Engine queue master:

  `$SGE_ROOT/default/spool/qmaster/messages`

- Sun Grid Engine execution daemon:

  `$SGE_ROOT/default/spool/`*`exec-hostname`*`/messages`

▼ To Verify Your Administrative Hosts

- **Type:**

```
# qconf -sh
```

▼ To Add Administrative Hosts

- **Type:**

```
# qconf -ah hostname
```

▼ To Obtain Current Status

- **Type:**

```
# qstat -f
```

Note – In the status display, `BIP` means that queue permits batch, interactive, and parallel jobs. Also, the status `au` means the execution host daemon (`execd`) is not successfully running and communicating with the `qmaster` process.

▼ To Start the Sun Grid Engine GUI

1. **Ensure that your `DISPLAY` environment variable is set appropriately.**
2. **Type:**

```
# qmon &
```

Installing Sun Shared Visualization 1.1 Software

The section describes how to install the Sun Shared Visualization 1.1 software and how to remove the software on Solaris and Linux systems. Instructions for installing and removing this software on Windows and Mac OS X clients are provided in the *Sun Shared Visualization 1.1 Software Client Administration Guide*.

Note – When installing the software onto a client system, the optional Sun Grid Engine supporting software is normally mounted from the grid's NFS server rather than installed on each client. Therefore, the `SUNWsge3D`, `SUNWsgear`, `SUNWsgeau`, `SUNWsgearsmr`, and `SUNWvglsr` packages are not required.

If you are installing the Sun Shared Visualization 1.1 software onto a Linux host, you might see the following error if using the automounter with default options, or you have `noexec` in the CD-ROM mount entry of the `/etc/fstab` file:

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

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

▼ To Install the Sun Shared Visualization 1.1 Software

1. **Decide what source to use for installing this software.**
   - If you are installing the software from a download directory, perform Step 2.
   - If you are installing the software from the CD-ROM, perform Step 3.
2. **Install this software from a download directory.**

   a. As superuser, change to that directory and extract each desired zip file.

```
# cd download directory
# unzip SharedVisualization_1.1.1_name.zip
```

Note – The version number (shown as *1.1.1* above and in TABLE 3-4) is release specific. You will find 1.1 or 1.1u1 in older releases.

See TABLE 3-4 for the names of each available zip file and the name of the directory where the expanded files will be installed:

TABLE 3-4 Operating Systems, Download Files, and Installation Directories

Operating System or Other	Item	Name
Documentation for any OS and platform	Download file	`SharedVisualization_1.1.1_docs.zip`
	Unzipped directory	`SharedVisualization_1.1.1_docs/Docs`
Solaris SPARC; Open Solaris or Solaris on x86 and x64	Download file	`SharedVisualization_1.1.1_solaris.zip`
	Unzipped directory	`SharedVisualization_1.1.1_solaris`
Linux (Red Hat, SuSE and Ubuntu)	Download file	`SharedVisualization_1.1.1_linux.zip`
	Unzipped directory	`SharedVisualization_1.1.1_linux`
Windows	Download file	`SharedVisualization_1.1.1_windows.zip`
	Unzipped directory	`SharedVisualization_1.1.1_windows`
x86 Mac OS X	Download file	`SharedVisualization_1.1.1_mac.zip`
	Unzipped files	`TurboVNC-`*version*`.dmg`* `VirtualGL-`*version*`.dmg`*

* The *version* number indicates a release of TurboVNC or VirtualGL software, not of the Sun Shared Visualization software

The directory structure is created and the files are extracted.

b. Change to the installation directory you selected from TABLE 3-4:

```
# cd SharedVisualization_1.1.1_name
```

c. Continue to Step 4.

3. Install this software from a 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 is running the volume manager, continue to Step b. Otherwise, type the following commands:

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

where *device* is:

OpenSolaris or Solaris – A path such as `/dev/dsk/c0t6d0s2`, obtained by running the `rmformat` command, but using `dsk` rather than `rdsk`

Linux – `/dev/cdrom`

Then continue to Step 4.

b. Change to the installation directory with a `cd` command.

The name of this directory varies.

- For OpenSolaris or Solaris, go to `/cdrom/ssv_1.1`:

```
# cd /cdrom/ssv_1.1*
```

- For Red Hat, go to `/cdrom/ssv_1.1` (or `/cdrom/SSV_1.1` or `/media/cdrom` or whatever name is provided by your version of the operating system):

```
# cd /cdrom/ssv_1.1*
```

- For SuSE, the dot in `1.1` might be replaced with an underscore character, so go to `/media/ssv_1_1` (or `/media/dvd` or whatever name is provided by your version of the operating system):

```
# cd /media/SSV1_1*
```

- For Ubuntu, go to `/cdrom` (or /media`/cdrom` or `/media/cdrom0` or whatever name is provided by your version of the operating system):

```
# cd /cdrom/
```

c. Continue to Step 4.

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:

```
application SUNWsge3D    Sun N1 Grid Engine Graphic Extensions
application SUNWsgearsmr Sun N1 Grid Engine Graphic Advance Reservations
application SUNWsgeau    Sun N1 Grid Engine Graphic Advance Reservations (Usr)

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

Note – The names of the packages shown here are for Solaris 10 versions.

6. **If you are installing on the NFS server for your Sun Grid Engine, answer** `y`**. Otherwise, answer** `n` **and go to** Step 8**.**

   - If you installed the Sun Grid Engine software into the default directory, you see this message:

```
Install script has determine that the Grid Engine Product install
directory as:
    /gridware/sge
```

   - If you installed the Sun Grid Engine software into a directory other than the default (`/gridware/sge`), or if the Sun Grid Engine software is not installed at all, you might see this message:

```
Unable to determine the installation directory for the Grid Engine Product.
Using default path of /gridware/sge.
```

   The script continues:

```
Press "Return" to accept the above path location or Enter the desired install
base directory path location [default install path: /gridware/sge, ? for
help] [?,q]
```

7. **If the value of** `$SGE_ROOT` **is not displayed or is different than** `/gridware/sge`**, type the new value and press Return.**

The script informs you:

```
This script is about to take the following actions:
- Install Sun Shared Visualization Software
- Install Optional Software (SUNWsge3D SUNWsgeau SUNWsgearsmr)
  using the installation path: /gridware/sge

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

Note – Optional software is only installed if you answered `y` at Step 6

8. **Press Return to continue installation.**

   The script begins installing required patches, packages, and optional software:

```
*** 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.

*** Installing selected optional software for Solaris...
Installing optional package(s):
       SUNWsge3D SUNWsgeau SUNWsgearsmr

Installation of <SUNWsge3D> was successful.
Installation of <SUNWsgeau> was successful.
Installation of <SUNWsgearsmr> was successful.

*** Installation complete.
```

Note – Optional software is only installed if you answered `y` at Step 6

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 9:52am.

9. **If your Sun Shared Visualization 1.1 server is also your Sun Ray server, refer to the section,** "Improving Sun Ray Image Quality at the Expense of Performance" on page 51**.**

10. **(Optional) If you are going to use the Sun Shared Visualization 1.1 software with Sun Grid Engine, see** "Adding Graphics to Sun Grid Engine" on page 64**.**

▼ To Remove the Sun Shared Visualization 1.1 Software

You might need to remove the Sun Shared Visualization 1.1 software in the future. This procedure explains how. Instructions for installing and removing this software on Windows clients are provided in the *Sun Shared Visualization 1.1 Software Client Administration Guide*.

1. **Take one of the following actions:**

   - If you are running the removal script from the Sun Shared Visualization 1.1 server, as superuser, type:

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

- If you are running the removal script from the CD-ROM, as a Solaris or Linux superuser, first insert, mount, and change directories (with the `cd` command) to the CD-ROM as done in Step 3 of "To Install the Sun Shared Visualization 1.1 Software" on page 43. Then type the appropriate one of these commands:

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

Or:

```
# 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:
 SUNWsge3D SUNWsgearsmr SUNWsgeau 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 Return 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:
          SUNWsge3D SUNWsgearsmr SUNWsgeau SUNWvglsr SUNWvgl SUNWtvnc SUNWvrpt
*** Removing old package(s)...

Removal of <SUNWsge3D> was successful.

Removal of <SUNWsgearsmr> was successful.

Removal of <SUNWsgeau> was successful.

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.

Improving Sun Ray Image Quality at the Expense of Performance

By default, the Sun Ray Image Transport uses DPCM compression with 16x chrominance subsampling to send images from the 3D application server to the Sun Ray client. These compression settings are designed to decrease bandwidth usage and to produce the maximum possible frame rate. However, these settings also produce noticeable compression artifacts, particularly on sharp lines or text.

Sun Shared Visualization 1.1.1 also supports sending uncompressed YUV images to the Sun Ray client, which increases network bandwidth usage by about 50% but which greatly improves the image quality of the Sun Ray Image Transport. Lossless image quality can be achieved if YUV encoding is used in conjunction with 1X chrominance subsampling, which is not the default. YUV encoding can be enabled in

the Sun Ray Image Transport by setting the `VGL_COMPRESS` environment variable to `sryuv` or passing an argument of `-c sryuv` to `vglrun`. For more information about VirtualGL configuration options, see the appendix "VirtualGL Reference" in the *Sun Shared Visualization 1.1.1 Software Client Administration Guide*.

Image quality can also be improved by using the X11 Image Transport, which is accomplished by setting the `VGL_COMPRESS` environment variable to `proxy` or passing an argument of `-c proxy` to `vglrun`. For performance reasons, this is not recommended if the Sun Ray server and the 3D application server are different machines. See Appendix A for more information.

The VirtualGL Sun Ray plug-in can be disabled by using the `-c proxy` option with `vglrun` or by setting the environment variable `VGL_COMPRESS` to `proxy`. See the appendix “VirtualGL Reference” in the *Sun Shared Visualization 1.1 Software Client Administration Guide*.

Alternatively, the VirtualGL Sun Ray plug-in can be disabled for all users on the server by removing the Solaris `SUNWvglsr` package or the Linux `VirtualGL-SunRay` package after installing the Sun Shared Visualization software. Once the plug-in is removed, VirtualGL will use the VGL Image Transport or the X11 Image Transport to send images to the Sun Ray server. The Sun Ray server will be responsible for re-encoding those images for transmission to the Sun Ray clients. This behavior can be advantageous in cases where the Sun Ray clients are located across a wide-area network. The Sun Ray server can use more advanced forms of image compression, such as wavelet compression, to send images to Sun Ray clients. These advanced forms of image compression perform much better on wide-area networks than the Sun Ray Image Transport.

CHAPTER 4

Configuration Information and Guidelines

This chapter provides configuration information for both OpenSolaris or Solaris servers and Linux based Sun Shared Visualization 1.1 servers. Topics include:

To configure a *client* system, see the *Sun Shared Visualization 1.1 Software Client Administration Guide*.

Configuration Overview Information

Configuration Process Overview

Shared Visualization 1.1 dramatically eases graphics host configuration, compared to earlier releases. In this release, you take the following steps to configure a host with a graphics accelerator to be a graphics server:

1. Decide which options you will use to configure VirtualGL.
2. Enable X11 forwarding for `ssh`.
3. Shut down the window system.

4. Invoke `vglserver_config`, then select a configuration option and answer questions presented for that option.
5. Restart the window system.
6. Verify the host as a graphics server for VirtualGL.
7. If this host will also be a Sun Grid Engine execution host:
   a. Configure Sun Grid Engine's graphics resources.
   b. Verify the host as a graphics execution host.

Granting Access to the 3D X Server

Tip – If you plan to use VirtualGL only with GLP, then you can skip this section.

VirtualGL requires access to the application server's 3D graphics accelerator so that VirtualGL can create off-screen pixel buffers (Pbuffers) and redirect the 3D rendering from applications into these Pbuffers. Accessing a 3D graphics accelerator on a Linux system, an OpenSolaris or Solaris x86 system, or a Solaris SPARC system without GLP requires going through an X server. On such systems, the only way to share the application server's 3D graphics accelerator among multiple users is to grant those users access to the "3D X server" (the X server attached to the application server's 3D graphics accelerator). Refer to the figures in Chapter 1.

Be aware that some security risks are associated with this sharing. Once a user has access to the 3D X server, nothing prevents that user from logging keystrokes or reading back images from the X display. Using `xauth`, one can obtain untrusted X authentication keys that prevent this behavior, but unfortunately, those untrusted keys also disallow access to the 3D hardware. Therefore, you must grant full trusted X access to any users needing to run VirtualGL. Unless you fully trust the users to whom you are granting this access, you should avoid logging in locally to the 3D X server, particularly as `root`, unless absolutely necessary.

Enabling X11 Forwarding for `ssh`

The server's SSH® daemon (`ssh`) should have the `X11Forwarding` option enabled. You configure this option in `sshd_config`. TABLE 4-1 lists the location of `sshd_config`, depending on your distribution of `ssh`.

TABLE 4-1 Locations of `sshd_config` According to SSH Distribution

SSH Distribution	Location of `sshd_config`
OpenSolaris or Solaris 10	`/etc/ssh`
Most Linux distributions	`/etc/ssh`
Blastwave	`/opt/csw/etc`
SunFreeware	`/usr/local/etc`

When `X11Forwarding` is enabled, its line in `sshd_config` is:

```
X11Forwarding    yes
```

The `UseLogin` option of SSH is incompatible, so that option must not be enabled in `sshd_config`. You can specifically disable `UseLogin` or simply not mention it in `sshd_config`.

Configuration Information for OpenSolaris or Solaris Servers

Before running `vglserver_config` to configure a Solaris SPARC graphics server, decide which of the following options you will choose:

- GLP
- GLX

An OpenSolaris or Solaris x86 or x64 system can use only GLX.

Setting Device Permissions

When a user logs into a Solaris host locally, the system automatically assigns ownership of the framebuffer devices to that user and sets the permissions for those devices to those specified in `/etc/logindevperm`. The default permissions disallow anyone from using the graphics accelerator devices except the user that is logged in.

In order to run VirtualGL, however, a user must have write access to the graphics accelerator devices as a shared resource. `vglserver_config` disables the login device permissions mechanism for the graphics accelerator devices. This command also sets the owner and group for these devices such that any member of the `vglusers` group (or optionally, all users) can write to the devices.

Note – If you opt to allow 3D graphics accelerator device access only to the `vglusers` group, then only users of this group will be able to run OpenGL applications locally on the graphics server, even if others can log in locally to the server.

Using GLP on Solaris SPARC Servers

A Solaris SPARC graphics server with Sun OpenGL 1.5 and an XVR-2500, XVR-1200, or XVR-600 graphics accelerator can be configured to use those devices through GLP without having to start a "3D X server" on the graphics accelerators. (See "GLP Access on Solaris SPARC Servers" on page 16 for more details.) If you have a graphics server with those characteristics, you can use `vglserver_config` to configure the server to use GLP mode exclusively.

Note – Sun OpenGL 1.5 is available at:
`http://www.sun.com/software/graphics/opengl`

Disabling the `XTEST` Extension

Unless absolutely needed, disable the `XTEST` extension. (For example, `XTEST` is disabled by Step 6 of "To Configure an OpenSolaris or Solaris Server to Grant Access to the 3D X Server" on page 59.)

Note – Disabling `XTEST` does not prevent a user from logging keystrokes or reading images from the X display. Disabling `XTEST` does prevent the user from inserting key and mouse events, thus possibly hijacking a local X session.

▼ To Configure a Solaris SPARC Server to Use VirtualGL With GLP and Without a 3D X Server

Use this procedure if you determined in "Using GLP on Solaris SPARC Servers" on page 56 that you will use GLP on your Solaris system.

1. **Log in as `root` and enter:**

```
# /opt/VirtualGL/bin/vglserver_config
```

A list of options is displayed.

2. **Select this option:**

```
Configure server for use with VirtualGL in GLP mode.
```

3. **Reply `y` or `n` to this question:**

```
Restrict framebuffer device access to vglusers group (recommended)?
[Y/n]
```

- **Yes –** Only users in the `vglusers` group can run OpenGL applications on the VirtualGL server. (If the `vglusers` group doesn't already exist, the `vglserver_config` script will create it.) This option limits the possibility that an unauthorized user can snoop a 3D framebuffer device and thus see or alter the 3D output of an application running in VirtualGL.
- **No –** Any authenticated user can run OpenGL applications on the VirtualGL server. If it is necessary to enable users outside of the `vglusers` group to log in locally to the server and run OpenGL applications, then this option must be selected.

4. **If you answered `y` in Step 3, edit the `vglusers` group in `/etc/group`.**

   If framebuffer device access will be restricted to the `vglusers` group, edit that entry to include `root` and any additional users. Any users that you add to `vglusers` at this time must log out and back in again before their new group permissions take effect.

5. **Edit the `/etc/dt/config/GraphicsDevices` file as necessary.**

   This file contains a list of paths to 3D framebuffer devices that you want to use with VirtualGL. Each path is on a separate line. For example:

   ```
   /dev/fbs/kfb0
   /dev/fbs/jfb0
   ```

6. **Verify that the system is ready to run VirtualGL.**

   a. **Log out of the server.**

   b. **Log back in to the server using SSH.**

   c. **Execute the following command in the SSH session:**

   ```
   /opt/VirtualGL/bin/glxinfo -d glp
   ```

   This command should output a list of visuals and complete with no errors.

7. **If you want VirtualGL to use GLP by default, add these lines to `/etc/profile`:**

   ```
   VGL_DISPLAY=glp
   export VGL_DISPLAY
   ```

   These lines cause VirtualGL to use the first device specified in `/etc/dt/config/GraphicsDevices` as the default rendering device. A user can override this default by setting `VGL_DISPLAY` in a startup script (such as `~/.profile` or `~/.login`) or by passing `vglrun` an argument of `-d` *device* when invoking VirtualGL.

▼ To Configure an OpenSolaris or Solaris Server to Grant Access to the 3D X Server

Use this procedure if you determined in "Using GLP on Solaris SPARC Servers" on page 56 that you will not use GLP on your Solaris SPARC system, or if your OpenSolaris or Solaris system does not have a SPARC processor. This procedure configures a VirtualGL server so that selected users can run VirtualGL, even if the server is currently at the login prompt.

1. **Shut down the display manager.**
   - On an OpenSolaris server running GDM, enter:

```
# scvadm disable gdm
```

   - On a Solaris 10 server running GDM, enter:

```
# scvadm disable gdm2-login
```

   - On a Solaris server running `dtlogin`, enter:

```
# /etc/init.d/dtlogin stop
```

2. **Log in as `root` from the text console (or remotely using `ssh`) and enter:**

```
# /opt/VirtualGL/bin/vglserver_config
```

   A list of options is displayed.

3. **Select this option:**

```
Configure server for use with VirtualGL in GLX mode.
```

4. **Reply `y` or `n` to this question:**

```
Restrict local X server access to vglusers group (recommended)?
[Y/n]
```

- **Yes –** Only users in the `vglusers` group can use VirtualGL. (If the `vglusers` group doesn't already exist, the `vglserver_config` script will create it.) This option is the most secure, since it prevents any users outside of the `vglusers` group from accessing and exploiting the 3D X server.
- **No –** VirtualGL can be used by any user that successfully logs into the VirtualGL server. Also, the 3D X server can be accessed and potentially exploited by any user who is logged into the VirtualGL server. If you choose this option, disable the `XTEST` extension, unless is it absolutely needed.

5. **Reply `y` or `n` to this question:**

```
Restrict framebuffer device access to vglusers group (recommended)?
[Y/n]
```

- **Yes –** Only users in the `vglusers` group can run OpenGL applications on the VirtualGL server. (If the `vglusers` group doesn't already exist, the `vglserver_config` script will create it.) This option limits the possibility that an unauthorized user can snoop a 3D framebuffer device and thus see or alter the 3D output of an application running in VirtualGL.
- **No –** Any authenticated user can run OpenGL applications on the VirtualGL server. If it is necessary to enable users outside of the `vglusers` group to log in locally to the server and run OpenGL applications, then this option must be selected.

6. **Reply `y` or `n` to this question:**

```
Disable XTEST extension (recommended)?
[Y/n]
```

- **Yes –** Disabling `XTEST` will prevent a user who has access to the 3D X server from inserting keystrokes or mouse events and thus highjacking local X sessions on that display. However, disabling `XTEST` will not prevent a user from logging keystrokes or reading images from the 3D X server.
- **No –** VNC requires `XTEST`, so if you need to attach a VNC server to the 3D X server, then you must leave `XTEST` enabled. (However, this action isn't needed when you're starting a new TurboVNC session unrelated to the server's local 3D X server.)

7. **If you answered `y` in Step 4 or Step 5, edit the `vglusers` group in `/etc/group`.**

   If framebuffer device access will be restricted to the `vglusers` group, edit that entry to include `root` and any additional users. Any users that you add to `vglusers` at this time must log out and back in again before their new group permissions take effect.

8. **Restart the display manager.**

   - On an OpenSolaris server running GDM, enter:

```
# scvadm enable gdm
```

   - On a Solaris 10 server running GDM, enter:

```
# scvadm enable gdm2-login
```

   - On a Solaris server running `dtlogin`, enter:

```
# /etc/init.d/dtlogin start
```

9. **Verify that the system is ready to run VirtualGL.**

   a. **Log out of the server.**

   b. **Log back in to the server using SSH.**

   c. **Execute one of the following command sequences in the SSH session:**

      - If you restricted X server access to the `vglusers` group, enter:

```
# /usr/openwin/bin/xauth merge /etc/opt/VirtualGL/vgl_xauth_key
# /usr/openwin/bin/xdpyinfo -display :0
# /opt/VirtualGL/bin/glxinfo -display :0
```

      - If you did not restrict X server access, enter:

```
# /usr/openwin/bin/xdpyinfo -display :0
# /opt/VirtualGL/bin/glxinfo -display :0
```

      Either command should output a list of visuals and complete with no errors. If `xdpyinfo` fails to run, then the permissions on display `:0` are too restrictive.

   d. **If you chose to disable the `XTEST` extension, check the output of `xdpyinfo` to verify that `XTEST` is not included in the list of extensions.**

Configuration Information for Linux Servers

This section explains how to configure a VirtualGL server such that select users can run VirtualGL, even if the server is currently at the login prompt.

▼ To Grant VirtualGL Access to the 3D X server on a Linux Server

1. **If the server is configured to boot into run level 5 (graphical login), temporarily shut down the X server as** `root`**. Type:**

```
# init 3
```

2. **Otherwise, log in as** `root` **from the text console.**
3. **Type:**

```
# /opt/VirtualGL/bin/vglserver_config
```

   A list of options is displayed.

4. **Select this option:**

```
Configure server for use with VirtualGL in GLX mode.
```

5. **Reply** `y` **or** `n` **to this question:**

```
Restrict local X server access to vglusers group (recommended)?
[Y/n]
```

   - **Yes –** Only users in the `vglusers` group can use VirtualGL. (If the `vglusers` group doesn't already exist, the `vglserver_config` script will create it.) This option is the most secure option, since it prevents any users outside of the `vglusers` group from accessing and exploiting the 3D X server.

- **No –** VirtualGL can be used by any user that successfully logs into the VirtualGL server. Also, the 3D X server can be accessed and potentially exploited by any user who is logged into the VirtualGL server. If you choose this option, disable the `XTEST` extension, unless is it absolutely needed.

6. **Reply `y` or `n` to this question:**

```
Restrict framebuffer device access to vglusers group (recommended)?
[Y/n]
```

- **Yes –** Only users in the `vglusers` group can run OpenGL applications on the VirtualGL server. (If the `vglusers` group doesn't already exist, the `vglserver_config` script will create it.) This option limits the possibility that an unauthorized user can snoop a 3D framebuffer device and thus see or alter the 3D output of an application running in VirtualGL.
- **No –** Any authenticated user can run OpenGL applications on the VirtualGL server. If it is necessary to enable users outside of the `vglusers` group to log in locally to the server and run OpenGL applications, then this is option must be selected.

7. **Reply `y` or `n` to this question:**

```
Disable XTEST extension (recommended)?
[Y/n]
```

- **Yes –** Disabling `XTEST` will prevent a user who has access to the 3D X server from inserting keystrokes or mouse events and thus highjacking local X sessions on that display. However, disabling `XTEST` will not prevent a user from logging keystrokes or reading images from the 3D X server.
- **No –** The VNC X extension requires `XTEST`, so if you need to attach a VNC server to the 3D X server, then you must leave `XTEST` enabled. (However, this isn't needed when you're starting a new TurboVNC session unrelated to the 3D X server.)

8. **If you answered `y` in Step 5 or Step 6, edit the `vglusers` group in `/etc/group`.**

   If framebuffer device access will be restricted to the `vglusers` group, edit that entry to include `root` and any additional users. Any users that you add to `vglusers` at this time must log out and back in again before their new group permissions take effect.

9. **As the `root` user, restart the X server. Type:**

```
# init 5
```

10. **Verify that the system is ready to run VirtualGL.**

    a. **Log out of the server.**

    b. **Log back in to the server using SSH.**

    c. **Execute one of the following command sequences in the SSH session:**

    - If you restricted 3D X server access to the `vglusers` group, type:

```
# xauth merge /etc/opt/VirtualGL/vgl_xauth_key
# xdpyinfo -display :0
# /opt/VirtualGL/bin/glxinfo -display :0
```

    - If you did not restrict 3D X server access, type:

```
# xdpyinfo -display :0
# /opt/VirtualGL/bin/glxinfo -display :0
```

    Either command should output a list of visuals and complete with no errors. If `xdpyinfo` fails to run, then the permissions on display `:0` are too restrictive.

    d. **If you chose to disable the `XTEST` extension, check the output of `xdpyinfo` to verify that `XTEST` is not included in the list of extensions.**

How This Method Works

This method calls `vglgenkey` from the display manager's startup script. `vglgenkey` calls `xauth` to generate an authorization key for the 3D X server and stores this key under `/etc/opt/VirtualGL`. The VirtualGL launcher script (`vglrun`) then attempts to read this key and merge the key into the user's `.Xauthority` file, granting the user access to the 3D X server. Using this method, you can control who has access to the 3D X server by controlling who has read access to the `/etc/opt/VirtualGL` directory.

Adding Graphics to Sun Grid Engine

This section describes how to add graphics resources to Sun Grid Engine. You must first install Sun Grid Engine and the Sun Shared Visualization 1.1 software before continuing with this procedure.

These steps are to be performed as the `sgeadmin` user on the queue master host, or on an administrative host that mounts `$SGE_ROOT` read-write.

▼ To Set the Variables

1. **Set `$SGE_ROOT` and set `PATH` to include Sun Grid Engine directories:**

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

   where `/gridware` is the base directory of your `$SGE_ROOT`.

2. **Assure that your `DISPLAY` environment variable is set and refers to the system whose X server keyboard you are using:**

```
% setenv DISPLAY myhost:0.0
```

   where *myhost* is the hostname of the X server, and `:0.0` identifies the X screen and display.

   If you access the server using `ssh -X`, `ssh` sets `DISPLAY` for you. However, the `ssh` tunnel is available only on that server host, not on all execution hosts in the grid.

▼ To Add Graphics to Sun Grid Engine

Note – If you are upgrading an existing Sun Shared Visualization 1.1 software installation, you only need to perform Step 1, Step 8, and Step 9.

---

1. **If the optional software was not already installed on the grid's NFS server, then, as superuser, install that software.**

   - On an OpenSolaris or Solaris NFS server, install the `SUNWsge3D` package into the `$SGE_ROOT` directory:

```
# pkgadd -d download-directory SUNWsge3D
```

Note – Ensure that your `$SGE_ROOT` value is your answer to the installation prompt, "`Please enter your SGE_ROOT directory`."

   - On a Linux NFS server, install the `sun-n1ge-3D.noarch.rpm` package into the `$SGE_ROOT` directory:

```
# rpm -iv /path-to-rpm-file/sun-n1ge-3D.noarch.rpm
```

2. **Set an administrative email for Sun Grid Engine so that all errors are reported through email.**

   a. **Type:**

```
% qconf -mconf
```

   This command starts your `$EDITOR` with a file containing configuration variables.

   b. **Add the email address for the `administrator_mail` configuration variable, then save and quit the file.**

3. **Add resource names to the Sun Grid Engine complex.**

   The complex is the vocabulary of variables that can be specified. Eight resources will be added in this step (in the 1.1.1 release): `graphics`, `dedicated_graphics`, `graphics_alone`, `maximum_graphics`, `headnode`, `chromium`, `sc_cols`, and `sc_rows`. (Some of those resources are used by Sun Scalable Visualization software.) TABLE 4-2 describes the four resources most important for Sun Shared Visualization.

TABLE 4-2 Key Resources Added to Sun Grid Engine Complex for Sun Shared Visualization

Resource Name	graphics	maximum_graphics	graphics_alone	chromium
Shortcut	gfx	maxgfx	alone	cr
Type	INT	INT	INT	INT
Relation	<=	<=	<=	<=
Requestable	YES	YES	YES	YES
Consumable	YES	NO	NO	YES
Default	0	0	0	0
Urgency	0	0	0	0

Further details on some of these resources are provided in "More Graphics Resource Allocation Information" on page 74.

Add resource names by running the `add_to_complex` script. As a Sun Grid Engine administrator (`sgeadmin`), type:

```
% cd $SGE_ROOT/graphics
% ./add_to_complex
```

The script adds the information in TABLE 4-2 to your Sun Grid Engine complex. The script reports if a resource already exists or is added. When the script is finished, it will prompt you to perform the next step in this procedure.

4. **Define which hosts have how many graphics resources available.**

   This step determines the maximum number of simultaneous graphics jobs that Sun Grid Engine could start on that host. For example, if your host has two graphics boards and the boards can accommodate three jobs each, your resources would be 2 x 3, for a total of 6.

   The easiest way to specify graphics resources is by using the `config_gfx` script.

   a. **As `sgeadmin` or `root` on each graphics execution host, type:**

```
% cd $SGE_ROOT/graphics
% ./config_gfx
```

   By default, the graphics server host is the host on which `config_gfx` is invoked. The default name of the queue to be configured is `all.q`. To use different values for either of these systems, type `config_gfx` with the `-h` (host) or `-q` (queue) option:

```
config_gfx [-h gfxServer] [-q queueName]
```

   b. **Respond to questions asked by the `config_gfx` script.**

   Your answers will determine values for some of the resources that were added in Step 3.

Question	Sun Grid Engine Resource	Comments
How many unique graphics boards?	`maximum_graphics`	
How many graphics jobs simultaneously?	`graphics`	2 or 3 applications per graphics accelerator is a good starting point.
Can graphics be dedicated to one user job? (Enter 1 for Yes, 0 for No)	`graphics_alone`	
Can this host be a Chromium headnode? (Enter 1 for Yes, 0 for No)	`chromium`	

   For GLP access to Solaris SPARC graphics devices, the count is of unique graphics boards, but for a host using GLX access, the count is the number of X screens. If a host is using Xinerama across multiple graphics devices, SGE cannot allocate those devices separately. See "GLX" on page 71 for more information about Xinerama and about configuring the list of these unique graphics devices or X displays.

The number of graphics jobs to allow simultaneously is of key importance. If this number is too high, graphics accelerator memory could be exhausted and performance will suffer greatly (for example, when applications page or swap their data onto the accelerator). If the number is too low, applications will need to wait until another application exits, even if an application is not actively using the graphics accelerator (for example, when the user of that application is away from the desk).

A good starting point is two or three applications per graphics accelerator. Allow more simultaneous jobs if the load is typically light users and data sets. Limit the number of simultaneous jobs if there are heavy users and data sets.

The following example of running `config_gfx` is for a Sun X4600 M2 server with two Nvidia QuadroPlex model 4s (a total of four graphics devices), which are not expected to run Chromium:

```
# config_gfx
How many unique graphics boards?
4
How many graphics jobs simultaneously?
8
Can graphics be dedicated to one user job? (Enter 1 for Yes, 0 for No)
1
Can this host be a Chromium headnode? (Enter 1 for Yes, 0 for No)
0
```

5. **Set the starter and epilog scripts for Sun Grid Engine's `all.q` cluster queue.**

   These scripts are hooks supported by Sun Grid Engine to provide queue-specific activity before and after a job runs.

   As `sgeadmin` or as `root`, type:

   ```
   % cd $SGE_ROOT/graphics
   % ./use_standard
   ```

   This action sets the starter and epilog scripts for all Sun Grid Engine queues.

6. **(Optional) Copy the `graphics/docs/README` file to a more user accessible location.**

Tip – The contents of the `README` file summarize Sun Grid Engine use. Edit the file to better describe your particular site, rename the file, and make the file available to users in `$SGE_ROOT`.

---

7. **Ensure that your** `DISPLAY` **environment variable refers to your 2D X server:**

```
% setenv DISPLAY myhost:0.0
```

where *myhost* is the hostname of the X server.

8. **Attempt to run a graphics job.**

This example submits to any Shared Visualization 1.1 graphics server:

```
% qrsh -b n /opt/VirtualGL/bin/vglrun -c proxy -spoil \
      $SGE_ROOT/graphics/RUN.glxspheres
```

where:

- `-b n` means the `vglrun` script is a Sun Grid Engine job script with options for your Sun Grid Engine job.
- `-c proxy` enables X11 Image Transport, so `vglclient` is not needed.
- `-spoil` disables frame spoiling, slowing rendering to display speed.

Note – This step uses the `-c proxy` option, which usually is not recommended due to its impact on performance. However, using this option here simplifies the verification process without ongoing impact.

The following example names a graphics execution host:

```
% qrsh -b n -q all.q@hostname /opt/VirtualGL/bin/vglrun -c proxy -spoil  \
      $SGE_ROOT/graphics/RUN.glxspheres
```

9. **Start** `$SGE_ROOT/graphics/empty_jobs` **from** `/etc/init.d/sgeexecd`**.**

`/etc/init.d/sgeexecd` is the Sun Grid Engine standard startup script. This script initiates shepherd processes. If these processes are shut down before the graphics jobs, you cannot reclaim the resources of those graphics jobs. To alleviate any possibility of this problem:

a. **Edit the** `/etc/init.d/sgeexecd` **file and around line 245, find** `$bin_dir/sge_execd`**.**

b. Insert the following text before that line:

```
pgrep -u sgeadmin sge_execd || $SGE_ROOT/graphics/empty_jobs
```

Replace `sgeadmin` in this line if your site uses a different SGE administrative login.

Note – Unless `/etc/init.d/sgeexecd softstop` was used, graphics jobs that are still running when `execd` is shut down lose their the `sge_shep` shepherd processes, so the epilog script is not started for the jobs. Consequently, the job allocator does not know about any graphics resources being consumed by such orphan jobs.

Note – You need to repeat this step if the Sun Grid Engine software is upgraded.

Sun Grid Engine Graphics Resources

Note – The steps referenced in this section pertain to the procedure "To Add Graphics to Sun Grid Engine" on page 65

You can control which graphics devices are used by Sun Grid Engine by creating or editing a world-readable local graphics configuration file `/etc/dt/config/GraphicsDevices` on any execution host. If the `GraphicsDevices` file is not present, only X server `:0.0` is used by GLX.

▼ To Create a `GraphicsDevices` File

This procedures also creates a directory for the `GraphicsDevices` file.

- **As superuser, type these commands:**

```
# mkdir -p /etc/dt/config
# touch /etc/dt/config/GraphicsDevices
# chmod 644 /etc/dt/config/GraphicsDevices
```

GLP

On a SPARC Solaris graphics server, `kfb` (XVR-2500) devices and `jfb` (XVR-1200 and XVR-600) devices can be used by VirtualGL through GLP. There is no need for a 3D X server on the devices. See "To Configure a Solaris SPARC Server to Use VirtualGL With GLP and Without a 3D X Server" on page 57. On such a host, each line of the graphics server's configuration file can be a device name followed by an optional maximum number of concurrent jobs for that device. (If no number is added, the default is that the device is used by only one job at a time.) For example:

```
# Device          NumberOfSimultaneousSGEJobs
/dev/fbs/jfb0
/dev/fbs/jfb1     2
/dev/fbs/jfb2     0
```

In this example, the host's `jfb0` device can support only one Sun Grid Engine graphics job. The `jfb1` device can only support up to two Sun Grid Engine graphics jobs. The `jfb2` device is not used for any Sun Grid Engine graphics jobs. (This device might be used by a local console user.)

GLX

Any UNIX host can be configured to start a "3D X server" on each graphics device. The `vglgenkey` technique of `vglserver_config` (described in "Configuration Information for Linux Servers" on page 62) will enable access to that display for VirtualGL users. If there are several graphics accelerators, the local configuration file can control allocation of jobs to these X displays or screens (for example, an x86 host with two devices used by screens `:0.0` and `:0.1`).

Xinerama

Xinerama is an extension to the X Window System that enables multiheaded X. When X is configured to use Xinerama, X can provide a user with one large virtual screen spread across two or more physical displays (also called *heads*). This configuration enables any application's window to be moved from one physical display to another, or for one window to span multiple displays. A similar effect can be produced, without using Xinerama, by a single graphics accelerator that is able to offer a single desktop across multiple monitor outputs. Use Xinerama with Sun Scalable Visualization Software on a host with multiple graphics accelerators that drive any portion of a group of displays (also called a *power wall*).

When multiple graphics devices are used with Xinerama, X provides only a single large, virtual screen, which is typically `:0.0`. Therefore, applications do not generally control the head on which a window or dialog will appear. VirtualGL uses only this single virtual X screen to provide remote users with GLX access to the graphics accelerator devices that drive the multiple heads. All remote visualization users share the first graphics accelerator, which causes resources on subsequent graphics devices to be underutilized.

Multiple Display X Without Xinerama

Without Xinerama, multiple graphics accelerators can be used as independent devices. In this case, the X desktop on each device is an independent desktop, but all desktops share the mouse and keyboard. Application windows started on one device must remain on that device. Those windows cannot be moved to a different device, nor can the windows span across multiple devices.

Because X treats each graphics device as a separate screen (typically, `:0.0` and `:0.1`), VirtualGL can use any of these X screens to provide remote users with GLX access to a graphics accelerator device. For remote visualization, you normally want to configure a single 3D X server to use all the graphics devices without Xinerama. Then configure Sun Grid Engine to allocate all graphics accelerators to remote visualization jobs, a configuration that provides load balancing.

Consequently, the host's `/etc/dt/config/GraphicsDevices` local configuration file might be:

```
# Display  NumberOfSimultaneousSGEJobs
:0.0        2
:0.1        2
```

`graphics` Resource Value

The number of graphics resources for each execution host (Step 4 in "To Add Graphics to Sun Grid Engine" on page 65) is the maximum number of concurrent graphics jobs Sun Grid Engine schedules for that host. The total of the maximum number of jobs on all graphics devices in the local configuration file should be no less.

Similarly, the execution host should have at least as many total slots as the total number of maximum jobs for concurrent jobs, if you want the execution host to allow that many concurrent graphics jobs. The Sun Grid Engine default for slots is the number of CPUs (cores) in the system when Sun Grid Engine's `install_execd` (Sun Grid Engine execution host configuration) script is run.

The configuration files are used by the graphics allocation script, `$SGE_ROOT/graphics/alloc` (or `alloc.debug`). You can study the script and comments, to adjust the script to suit your needs. Make a copy of your changes, so that your customizations to the script can be reintegrated in the event of a software upgrade.

Advanced Allocation Control

Example of Reconfiguration

A user might *demand* a certain number of graphics boards for a job. This is a *hard limit*. If not possible, the job does not run. The hard limit is specified (for example, Step 8 of "To Add Graphics to Sun Grid Engine" on page 65), with `-l gfx=`*value*.

A user might also *request* a desired (maximum) number of graphics boards, which Sun Grid Engine calls a *soft limit*. In this situation, a queued job waits for a time when more resources are available. An interactive job is immediate and fails. A soft limit needs the `-soft` introduction, and also must use a different resource, `maximum_graphics` (shorthand: `maxgfx`). This situation is due to Sun Grid Engine restrictions.

A user can start `qsub` or `qrsh` specifying both hard quantities of necessary resources and soft quantities of desired resources. For example:

```
% qsub -hard -l gfx=1 -soft -l maxgfx=4 mygraphicsprogram
```

In this example, the job requests four graphics devices, but demands at least one. If two devices are assigned, the `VGL_DISPLAY` value in the environment of the job contains two words. Each word could be a graphics device name (on a SPARC Solaris host using GLP, such as `/dev/fbs/kfb0`;) or an X display name (such as `:0.0` or `:0.1`).

VirtualGL itself only uses the first device (or display) in the `VGL_DISPLAY` environment variable value. Allocating multiple devices is of value only if the job divides work among multiple processes, using one device or display value for each process.

More Graphics Resource Allocation Information

The following tables provide more information about the Sun Grid Engine integers and environment variables that control allocation of graphics resources:

- TABLE 4-3, `graphics` integer
- TABLE 4-4, `maximum_graphics` integer
- TABLE 4-5, `graphics_alone` integer
- TABLE 4-6, `graphics_include` variable
- TABLE 4-7, `graphics_exclude` variable

TABLE 4-3 `graphics` Integer

`graphics`	Shorthand: `gfx`	INT	Requestable	Consumable	Default: 0 (no graphics)
To a User	The number of graphics resources the job needs.				
To the SysAdmin	The maximum number of graphics resources Sun Grid Engine allocates to all simultaneous jobs. This number should be no larger than the total of job counts in the `/etc/dt/config/GraphicsDevices` files. A system administrator can control this resource by execution host or by queue.				
Comments	A user can specify both a minimum (required) graphics resource count and a desired (maximum) graphics resource count. Sun Grid Engine does not schedule the job until Sun Grid Engine determines that at least the minimum can be allocated, and then allocates up to the maximum and decreases the number left for other jobs correctly.				
Example 1	`qrsh -v DISPLAY -l gfx=1 my_app`				Requires graphics.
Example 2	`qrsh -v DISPLAY -l gfx=2 job_needing_2_resources`				Requires 2 graphics devices.
See Also	Sun Grid Engine disallows a soft (desired) limit for a consumable resource such as graphics. Use `maximum_graphics` instead.				

TABLE 4-4 `maximum_graphics` Integer

`maximum_graphics`	Shorthand: `maxgfx`	INT	Requestable	Not consumable	Default: 0 (no graphics)
To a User	The maximum number of graphics resources the job desires. This method is a way for a user to describe a soft limit for the graphics resource.				
To the SysAdmin	The maximum number of graphics resources a user can express as a desire. A system administrator can control the resource by execution host or by queue.				

TABLE 4-4 `maximum_graphics` Integer *(Continued)*

`maximum_graphics`	Shorthand: `maxgfx`	INT	Requestable	Not consumable	Default: 0 (no graphics)
Comments	A user can specify both a minimum graphics resource count using `gfx` and a desired graphics resource count using `maxgfx`. Sun Grid Engine does not schedule the job until Sun Grid Engine determines that at least the minimum can be allocated, and graphics allocation increases to the maximum number of graphics resources.				
Warning	When `maximum_graphics` exceeds `graphics`, a job can be allocated more graphics resources than Sun Grid Engine determines are consumed. This situation can lead to a case where Sun Grid Engine schedules a later job for execution on that host, but that job cannot be allocated as many graphics resources as the job requires. Such a job continually attempts to restart unless the administrator sets `FORBID_RESCHEDULE` in the Sun Grid Engine configuration.				
Example 1	`qsub -v DISPLAY -l gfx=2,maxgfx=4 two_to_four`				Requires 2 graphics devices, but desires 4. 2, 3, or 4 could be allocated, yet Sun Grid Engine knows about 2.

TABLE 4-5 `graphics_alone` Integer

`graphics_alone`	Shorthand: `alone`	INT	Requestable	Not consumable	Default: 0 (no graphics)
To a User	Set to 1 to indicate that you want dedicated graphics devices. By default, graphics devices could be shared with other jobs.				
To the SysAdmin	Set to 1 to enable a user to require dedicated graphics devices. A system administrator can control this resource by execution host or by queue.				
Comments	A user requesting multiple graphics resources using `gfx=`*N* (or `gfx=1` and `maxgfx=`*N)* could be allocated the same graphics devices multiple times if `graphics _alone` is not used. For example, `/dev/fbs/kfb0 /dev/fbs/kfb1 /dev/fbs/kfb0`. If `graphics_alone` is used, only unique devices are allocated.				
Example 1	`qsub -v DISPLAY -l gfx=1,gfx_alone=1 will_not_share`				Require a dedicated graphics device.
Example 2	`qrsh -v DISPLAY -l gfx=2,gfx_alone=1 two_dedicated_cards`				Requires 2 dedicated graphics devices.

TABLE 4-6 `graphics_include` Variable

`graphics_include`	No shorthand	Environment variable	Requestable using -v	Not consumable	Default: `""` (that is, all graphics devices in the `GraphicsDevices` file)
To a User	List of filenames or patterns of acceptable graphics device names. By default, any graphics devices in the `/etc/dt/config/GraphicsDevices` file could be allocated.				
To the SysAdmin	A system administrator can control devices that a user could be allocated by editing the `/etc/dt/config/GraphicsDevices` file or by putting the `-v graphics_include` option in an `sge_request` file for a Sun Grid Engine cell.				
Comments	`graphics_include` value can be a device name pattern, a list of device names, or a list of patterns. See examples. Note that patterns and multiple words must be quoted.				
Warning	If no devices match the pattern, a job enters the Error state. That is, `qstat -f` shows the job pending with status `E`. If the `GraphicsDevices` file does not exist, `VGL_DISPLAY` is `""` regardless of this environment variable.				
Example 1	`qsub -v DISPLAY -l gfx=1 -v graphics_include= /dev/fbs/kfb0 must_be_kfb0`				Require the named graphics device.
Example 2	`qrsh -v DISPLAY -l gfx=2 -v graphics_include= /"*kfb*" must_be_kfb_devices`				Requires 2 KFB graphics devices.
Example 3	`qrsh -v DISPLAY -l gfx=2 -v graphics_include= "*kfb[01] *jfb[01]" must_be_these_devices`				Requires 2 graphics devices matching a pattern shown.
See Also	`graphics_exclude`				

TABLE 4-7 `graphics_exclude` Variable

`graphics_exclude`	No shorthand	Environment Variable	Requestable using -v	Not consumable	Default:"" (no graphics device is excluded)
To a User	List of filename patterns of unacceptable graphics device names. By default, no graphics devices are excluded.				
To the SysAdmin	A system administrator can control devices that users could be allocated by editing the `/etc/dt/config/GraphicsDevices` file or by putting `-v graphics_include` option in an `sge_request` file for a Sun Grid Engine cell.				
Comments	`graphics_exclude` value can be a device name pattern, a list of device names, or a list of patterns. See examples. Note that patterns and multiple words must be quoted.				
Warning	If no devices are acceptable after exclusion, a job enters the Error state. That is, `qstat -f` shows the job pending with status `E`. If the `GraphicsDevices` file does not exist, `VGL_DISPLAY` is `""` regardless of this environment variable.				
Example 1	`qsub -v DISPLAY -l gfx=1 -v graphics_exclude= /dev/fbs/kfb0 must_not_be_kfb0`				Reject the named graphics device.
Example 2	`qrsh -v DISPLAY -l gfx=2 -v graphics_exclude= "*kfb*" must_not_be_kfb_devices`				Refuse KFB graphics devices.
Example 3	`qrsh -v DISPLAY -l gfx=2 -v graphics_include= "*kfb*, graphics_exclude=*kfb[01]" not_kfb0_nor_kfb1`				Requires KFB device, but neither `kfb0` or `kfb1`.
See Also	`graphics_include`				

▼ To Enable Graphics Allocation Logging

The graphics allocation called by the starter script on a graphics server host attempts logging of which users use how many graphics devices at what start and finish times, using the system logger. By default, these messages are `local0.info` messages and are discarded by the Solaris logger.

Follow this procedure to configure system logging to save the logging messages in `/var/adm/messages`.

1. **Review the** `syslog.conf(4)` **man page.**
2. **Open the** `/etc/syslog.conf` **file in an editor.**
3. **Search for the** `/var/adm/messages` **line following** `mail.crit`**.**
4. **Add** `;local0.info` **to that line.**

   For example:

```
*.err;kern.debug;daemon.notice;mail.crit;local0.info        /var/adm/messages
```

vglrun Interposing

When a job uses gfx=1 (or more) resources, Sun Grid Engine allocates one or more graphics accelerators. Sun Grid Engine also sets the graphics accelerator's device name into the VGL_DISPLAY environment variable used by vglrun.

Such a job can interpose on an application by starting the job with vglrun or /opt/VirtualGL/bin/vglrun if not in the users' $PATH.

If qsub starts vglrun directly, qsub requires the full path. For example:

```
% qsub /opt/VirtualGL/bin/vglrun -c proxy -spoil myGraphicsScript
```

Note – The arguments in this example are for vglrun, not Sun Grid Engine. Any #$ comments in the *myGraphicsScript* are not seen by Sun Grid Engine. However, the #$ comments within the vglrun script itself are seen by Sun Grid Engine.

vglrun can also start an executable. For example:

```
% qsub /opt/VirtualGL/bin/vglrun -spoil /opt/VirtualGL/bin/glxspheres
```

Also, a Sun Grid Engine job script can start vglrun when ready to run the application. 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" "$@"
```

VirtualGL cannot use multiple graphics accelerators, so you cannot specify `gfx` any greater than `1`, nor even configure `maxgfx`. To do so would consume resources without a performance improvement. You might want to specify `gfx` greater than `1` when your job requires several graphics accelerators concurrently, yet separately.

VirtualGL With TurboVNC

You might want Sun Grid Engine to allocate a graphics accelerator and start a TurboVNC server session to use the graphics accelerator. For example:

```
% qsub -l gfx=1 /opt/TurboVNC/bin/vncserver
```

When a shell in the TurboVNC session is ready to start an application, start `vglrun` from within a terminal on the TurboVNC session. For example:

```
% vglrun mygraphicsprogram
```

By writing a specialized script, Sun Grid Engine resources and `VGL_` environment variables can be set at run time. You can see this situation in the `$SGE_ROOT/graphics/RUN.vncserver` script.

Stereographic Support

If you need the server to support quad-buffered stereographic display for remote VirtualGL clients, read the requirements in the "Verifying Advanced Feature Support" section of Appendix A in the *Sun Shared Visualization 1.1 Software Client Administration Guide*.

▼ To Determine if a Server Has a Suitable Visual for Quad-buffered Stereographic Rendering

1. **Type one of the following on the VirtualGL server:**
   - On a Solaris server using GLP, type:

```
/opt/VirtualGL/bin/glxinfo -d glp-device -v
```

   - On a Linux server or an OpenSolaris or Solaris server using GLX, type:

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

2. **In the output, see if one or more of the visuals says `stereo=1` and lists `Pbuffer` as one of the Drawable Types.**

   This output is an indicator that the server is suitable.

▼ To Verify Client Visuals

Now you need to determine whether the X display on the client has a suitable visual for quad-buffered stereographic rendering, transparent overlays, or Pseudocolor.

1. **On the VirtualGL server, type:**

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

2. **In the output, look for the following:**
   - To be able to use quad-buffered stereo, one or more of the visuals should say `stereo=1`.
   - To be able to use transparent overlays, one or more of the visuals should say `level=1`, should list a `Transparent Index` (rather than list `Opaque`), and should have a class of `PseudoColor`.
   - To be able to use pseudocolor (indexed) rendering, one of the visuals should have a class of `PseudoColor`.

Unconfiguring the VirtualGL Server

You can use the `vglserver_config` script to restore the unshared, secure default settings that do not allow VirtualGL access. However, unconfiguring the server does not remove the `vglusers` group or the `/etc/dt/config/GraphicsDevices` file.

Both of the options described in the following procedure will restore the framebuffer device permissions to their default. The default is that the framebuffer devices can be accessed only by `root` or the user currently logged into the system locally.

▼ To Unconfigure the VirtualGL Server

1. **Shut down the display manager.**

- On an OpenSolaris server running GDM, type:

```
# scvadm disable gdm
```

- On a Solaris 10 server running GDM, type:

```
# scvadm disable gdm2-login
```

- On a Solaris server running `dtlogin`, type:

```
# /etc/init.d/dtlogin stop
```

- On a Linux server, type:

```
# init 3
```

2. **Log in as `root` from the text console (or remotely using `ssh`) and type:**

```
# /opt/VirtualGL/bin/vglserver_config
```

A list of options is displayed.

3. **To unconfigure GLX mode, select this option:**

```
Unconfigure server for use with VirtualGL in GLX mode
```

This option removes any shared access to the 3D X server, preventing VirtualGL from accessing the 3D hardware in that manner. This option also reenables the `XTEST` extension on the 3D X server.

4. **To unconfigure GLP mode, select this option:**

```
Unconfigure server for use with VirtualGL in GLP mode
```

5. **Restart the display manager as described in** Step 8**,** "Restart the display manager." on page 61**.**

Configuration Troubleshooting

Note – To resolve problems involving the proper use of `vglconnect`, refer to the *Sun Shared Visualization 1.1 Software Client Administrator Guide*.

- When using VGL Image Transport, `vglrun` requires that the `vglclient` program is already running on your client. If not, and you do not pass the `VGL_COMPRESS=proxy` environment variable or the `-c proxy` option to `vglrun`, `vglrun` immediately exits with a `Connection refused` error.

  The `vglclient` program is normally started implicitly by `vglconnect`.

- VirtualGL might fail with a message such as No GLP devices are registered. This error is usually caused when VirtualGL is being started in GLP mode, and either the `/etc/dt/config/GraphicsDevices` file does not exist on the graphics server or the current user account does not have permission to read that file. Check if one or more of the devices specified in `/etc/dt/config/GraphicsDevices` is not a valid framebuffer device.

- VirtualGL might fail with a message such as `Could not open display`. This error is usually caused by one of these conditions:
  - The 3D X server running on the graphics server has not been configured to allow access to VirtualGL users.
  - The current user account is not in the `vglusers` group on the graphics server.
  - There is no 3D X server running on the graphics server.
  - The graphics server is configured only for use with GLP, but the `vglrun` command did not specify the `-d glp` argument, nor was the `VGL_DISPLAY` environment variable set to `glp` or to a valid GLP framebuffer device.

`vglrun` Issues With Set-UID Programs and Scripts

The `vglrun` script can be used to launch either binary executables or shell scripts. When you use `vglrun` to run a shell script, the VirtualGL faker library is preloaded into every executable that the script launches. If the script invokes any executables that are `setuid` or `setgid`, the operating system might refuse to load VirtualGL into those executables because you are attempting to preload a library (VirtualGL) that is not in a secure path. The error message depends on the operating system.

- On the OpenSolaris or Solaris OS, the message might be:

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

  The Solaris OS allows preloading into a `setuid` or `setgid` process only if the library is in a directory recognized as containing only secure libraries.

- On Linux, the message might be:

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

  Linux allows preloading a library into a `setuid` or `setgid` process only if the library is also in `set-user-ID`. Linux may also require that the a library is in a standard search directories such as `/lib`, `/usr/lib`, and `/usr/lib64`.

The following sections describe alternative ways to work with this restriction.

Configure the Operating System to Consider VirtualGL to Be Secure

A system administrator can configure the operating system to consider the VirtualGL library to be secure Be aware of the security ramifications before you run these commands. You are essentially telling the operating systems that you trust the security and stability of the VirtualGL code as much as you trust the security and stability of the operating system. If you decide to use this alternative, the details depend on the operating system:

- On an OpenSolaris or Solaris system, superuser can use the `crle` utility to add other directories to the operating system's list of secure paths. For VirtualGL, type these commands as superuser:

```
# crle -u -s /opt/SUNWvgl/lib
# crle -64 -u -s /opt/SUNWvgl/lib/64
```

- On Linux, as superuser, use the chmod command to enable the `setuid` and `setgid` bits of the VirtualGL faker library, which is already installed in standard system directories::

```
# chmod +s /usr/lib/librrfaker.so /usr/lib64/librrfaker.so
```

Edit the Application Script To Control Preloading

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 the OpenSolaris or Solaris OS 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 executable. 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 not allow the user invoking `my_vgl_script` to provide `vglrun` options on the command line.

- 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 the OpenSolaris or Solaris OS, 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
```

Sun Grid Engine Graphics Extensions Troubleshooting

- There are two versions of the Sun Grid Engine graphics scripts. The two versions behave identically except for the following differences:
  - `use_debug` causes many messages to be saved in the job's `stdout` stream.
  - `use_standard` writes only minimal messages into the job's `stdout` stream.
- The `ls_jobs` script (`$SGE_ROOT/graphics/ls_jobs`) lists active jobs or jobs whose graphics usage has not been cleaned up by the `epilog` script. Comparing the output of `ls_jobs` to the output of `qstat -f` can help you determine if there are any jobs that have terminated, but have left graphics job files behind.
- The `rm_jobs` script (`$SGE_ROOT/graphics/rm_jobs`) could be started by `root` to clean up after a completed graphics job whose `epilog` did not do so. The `rm_jobs` script is started with a list of jobIDs (not `job.`*jobID* filenames). For example:

```
% rm_jobs 100 101 102
```

CHAPTER 5

Advance Reservation

Advance Reservation (AR) is a feature of some queuing software systems, but this feature is not 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.) AR schedules compute and visualization resources when the computer resources and the people to use the resources are both available. Reservations must not be scheduled to conflict with each other (by oversubscribing available resources), nor with other Sun Grid Engine uses of the same resources.

This chapter details information the system administrator needs to know about AR. The *Sun Shared Visualization 1.1 Software Client Administration Guide*, 820-3257, contains information for the end user.

Topics in this section include:

Advance Reservation Overview

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 Sun Grid Engine's `qsub` command) right after the reservation is confirmed, or anytime before the end of the reserved period.

Implementing Advance Reservations outside of Sun Grid Engine requires creating a dynamic Sun Grid Engine queue to represent each confirmed reservation. Resources are allocated to the reservation's queue by temporarily removing the resources from the execution host's generic queue (such as `all.q` or `interactive`). Resources are removed in advance of the reservation, so that non-AR jobs that use the resources are finished prior to when the reservation is scheduled to start.

Architecture of Advance Reservation Facility

The Advance Reservation package has two kinds of programs:

- An AR server

  The AR server assures that reservations consume only enabled resources. This server assures that confirmed reservations' resources are available to the users. For each confirmed reservation, the server dynamically creates a Sun Grid Engine queue that becomes active (that is, the queue's jobs can run) at the reservation time. The server runs as the Sun Grid Engine administrator (`sgeadmin`) on a Sun Grid Engine administration host.

- An AR client

  The AR client is used by any Sun Grid Engine user to create, list, and delete reservations. The client communicates with the AR server. The AR client exists in two forms. The Reserve client is a simple command-line program. The Reserve GUI is a graphical user interface that eases use.

Additional components involved in AR, shown in the architectural diagram (FIGURE 5-1), are:

- The reservation database

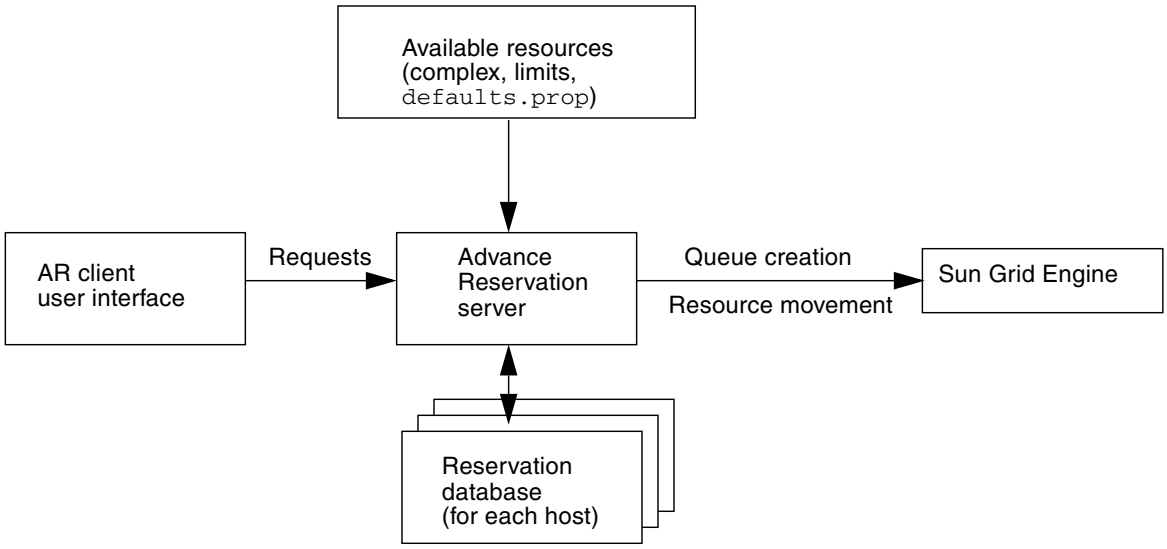
  This database is currently a Berkeley database, which is used only in primitive ways to make reservations persistent.

- Sun Grid Engine

  This application is the software that actually allocates resources to jobs, including jobs submitted against the reservation's queue.

FIGURE 5-1 shows the implementation architecture.

FIGURE 5-1 Advance Reservation Architecture



The system running the AR server and all systems running the AR client require a Java™ Runtime Environment (JRE™) software. The earliest version to support Advance Reservation is JRE 1.5 (known as Java 5).

Advance Reservation File Structure

The Advance Reservation package installs an `ar` subdirectory under the Sun Grid Engine `SGE_ROOT`. The default `SGE_ROOT` is `/gridware/sge`. If your `SGE_ROOT` differs from the default, you must set the `SGE_ROOT` environment variable, typically by sourcing the Sun Grid Engine `/gridware/sge/default/common/settings.csh` or `/gridware/sge/default/common/settings.sh` file.

Under `$SGE_ROOT/ar` you find these components:

TABLE 5-1 Directory Tree Under `$SGE_ROOT/ar`

Directory or Filename	Description or Comment
`bin/`	
`SERVER`	Script that starts the Advance Reservation server.
`runar`	Script to set up the necessary environment and then start Java™. This script might need editing for purposes such as: • Indicating the correct location of a Java that is at least 1.5 • Reflecting the `$SGE_ROOT/ar` location. A copy of this script is needed by users to start the AR client easily.
`config/`	The Advance Reservation server is configured using these files.
`queue.template`	This file is the basis for creation of a Sun Grid Engine dynamic queue to represent a reservation. This file is similar to the output of the Sun Grid Engine `qconf -sa all.q@‘hostname‘` command, but with spots ready to be replaced. You can edit this file to be more similar to your output of that command.
`users.template`	This file is the basis for creation of a Sun Grid Engine user list, and enables only the user creating a reservation to use the reservation. Adding users to this template (prior to `#Users#`) enables all such users to submit jobs to any reservation.

`defaults.prop`	A Java properties file that provides the execution host name and domain, the generic queue name (default `all.q`), and some configuring durations. For example:
	`#Advance Reservation Configuration Properties`
	`AdvanceReservation.ServerHost=my1`
	`AdvanceReservation.ServerPort=6789`
	`AdvanceReservation.ExecutionHostList=my1 my2 another`
	`AdvanceReservation.ExecutionHostDomain=my.company.com`
	`AdvanceReservation.GenericQueueName=all.q`
	`AdvanceReservation.MaximumNonreservedJobDuration=2\:0\:0`
	`AdvanceReservation.FinishToDeleteQueueDuration=12\:0\:0`

`complex`	Similar to `qconf -sc` output, this file contains the vocabulary of resources that can be reserved. Only integer consumable resources are currently supported. A resource could be required (as in Sun Grid Engine), meaning a reservation request would need to specify a value for that resource.
`limits`	Similar to `qconf -sq all.q` output, this file gives a maximum value of a resource that can be allocated by the AR server. For example, a host with 10 graphics resources might enable at most six to be used for AR, keeping four for temporary use (through Sun Grid Engine or outside of Sun Grid Engine entirely).
`lib/`	Directory of files needed by the Advance Reservation server and clients.

TABLE 5-1 Directory Tree Under `$SGE_ROOT/ar` *(Continued)*

Directory or Filename	Description or Comment
`je.jar`	Berkeley Database Java Edition (needed by server).
`*.jar`	Additional Java archives needed by the Advance Reservation server and clients.
`*.perl`	Scripts used by server (in preparing files for Sun Grid Engine).

Planning Configuration of Advance Reservation

Specifying a Nondefault `SGE_ROOT`

If during installation or configuration you choose a different location for `SGE_ROOT` than the default of `/gridware/sge`, and if the Solaris software package `SUNWsgeec` is not installed on the Advance Reservation server host, the Advance Reservation facility needs your `SGE_ROOT` value. You must edit several files to include the nondefault location so that the Sun Grid Engine and Advance Reservation feature function properly.

▼ To Edit the Files to Match a Nondefault `SGE_ROOT`

1. **Install the Sun Grid Engine software and optionally configure for Advance Reservation.**
2. **As superuser, use an editor to edit the following files:**
   - `$SGE_ROOT/ar/bin/runar`
   - `$SGE_ROOT/ar/bin/SERVER`
   - `/lib/svc/method/n1gear` (on a Solaris 10 or later system)
   - `/etc/init.d/sgear` (on all other systems)
3. **Within the files, locate each occurrence of the string:**

```
/gridware/sge
```

4. **Replace each occurrence with:**

/your-sge-root-path

where *your-sge-root-path* is your specific Sun Grid Engine root path.

Determining a Maximum Nonreserved Job Duration

If a reservation relies on certain resources being available at its start time *T*, no job can start (on the same host as the reservation) shortly before *T* using those resources and still be using them at time *T*. However, a job could start shortly before *T* using additional resources not required by the reservation.

`defaults.prop` contains a Java property called `AdvanceReservation.MaximumNonreservedJobDuration`. The value of this property is a duration. The default value is `2\:0\:0`, which means 2 hours, 0 minutes, and 0 seconds.

This property determines the amount of time preceding any existing reservation during which the AR server will not allow another job to start running if that job would call for the reserved resources. That duration prior to the job start, the reserved resources will be set aside for the reservation by removing them from the generic queue given by the Java property `AdvanceReservation.GenericQueueName` (the default value is `all.q`). These reserved resources will be released by the reservation at the end of the reservation. The released resources are returned to the generic queue, unless they are already known to be needed soon afterwards by a subsequent reservation.

To guarantee the reservation, you must preclude jobs started prior to the resource reservation (that is more than `AdvanceReservation.MaximumNonreservedJobDuration` before the reservation start time *T* is still running at time *T*). To achieve this, the grid administrator should assure that jobs specify a maximum runtime (wall clock time) limit no greater than the duration of the `AdvanceReservation.MaximumNonreservedJobDuration` property.

The maximum runtime limit is specified using the `h_rt` resource (whose values are also in the *hours*`:`*minutes*`:`*seconds* format). You specify the maximum `h_rt` resource limit in the cluster-wide `sge_request` file. (You also can place a `.sge_request` file in the current working directory or in `$HOME`.) For the default cell, this file is `$SGE_ROOT/default/common/sge_request`.

The specification of a maximum `h_rt` equal to 2 hours would look like:

```
-l h_rt=2:0:0
```

Note that an effective specification does not start with the # that appears in the comments already in the file.

Initial Configuration of Advance Reservation

The first time the Advance Reservation server runs, the server creates initial versions of the configuration files described in TABLE 5-1. The initial configuration enables Advance Reservation on all execution hosts in the grid of the server host. Initially, the limits file knows only of graphics resources that were assigned using Sun Grid Engine to a specific queue (for example, `all.q@mygraphicsserver`). The file does not know of those resources allocated to an execution host. To correct this situation, perform the procedure for your respective operating system.

▼ To Perform Initial Configuration for Solaris 10 and Later Operating Systems

1. **As superuser, configure the Advance Reservation server with the Sun Grid Engine administrative user.**

   For example:

   ```
   # svccfg -s n1ge_ar setprop config/admin_user= astring: sgeadmin
   ```

   Replace `sgeadmin` with your Sun Grid Engine administrator username, if different.

2. **Start the Advance Reservation service by typing:**

   ```
   # svcadm -v enable n1ge_ar
   ```

   The service is started and configured to start whenever the host reboots.

   The files described in TABLE 5-1 are created.

3. **Edit the `$SGE_ROOT/ar/config/limits` file to reflect any resources that are assigned to an execution host, rather than to a queue.**

4. **If any edits were performed in Step 3, have the Advance Reservation service reread its configuration by typing:**

```
# svcadm -v refresh n1ge_ar
```

The output of the AR server is saved in a log file named:
`/var/svc/log/network-n1ge_ar:default.log`

▼ To Perform Initial Configuration for Solaris 9 and Earlier and Linux Operating Systems

1. **As superuser, open the `/etc/init.d/sgear` script in an editor.**

2. **Set the `AR_USER` variable to your Sun Grid Engine administrator username.**

For example:

```
AR_USER=sgeadmin
```

Replace `sgeadmin` with your Sun Grid Engine administrator username, if different.

3. **Save and close the file.**

4. **Start the Advance Reservation service by typing:**

```
# /etc/init.d/sgear start
```

The files described in TABLE 5-1 are created.

5. **Edit the `$SGE_ROOT/ar/config/limits` file to reflect any resources that are assigned to an execution host, rather than to a queue.**

Restart the Advance Reservation service by typing:

```
# /etc/init.d/sgear restart
```

The output of the AR server is saved in a log file named:
`/var/tmp/ARS.$$` (*$$* is the process ID for the script).

Using Advance Reservation

Starting an AR Server or Client

To start the AR server or client, the `bin/runar` script is used. Normally, the server startup is performed by the RC script `/etc/init.d/sgear`.

The AR server must run on a Sun Grid Engine administrative host. The server runs as the Sun Grid Engine administrator (`sgeadmin`), and must have access to the Sun Grid Engine executables.

▼ To Manually Start the Advance Reservation Script

- **Type:**

```
myserver% /gridware/sge/ar/bin/runar [arguments]
```

If you have configured your `$SGE_ROOT` variable to something other than the default, you can also type:

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

For more simplicity, you can alias the complete path to the `runar` script to a single command. For example:

```
myserver% alias advance $SGE_ROOT/ar/bin/runar
```

Using an AR Client

Use of the Advance Reservation client is described in the *Sun Shared Visualization 1.1 Software Client Administration Guide*, 820-3257.

Reservation States

The following table lists the states a reservation normally passes through, in sequence:

TABLE 5-2 Reservation States

Reservation State	Description
`Specified`	Minimal user data has been specified for this reservation request.
`Confirmed`	Reservation request granted (the reservation is compatible with confirmed reservations).
`QueueMade`	A Sun Grid Engine queue has been created (with resources) for the reservation. A reservation should move to this state immediately after the server confirms the reservation. Sun Grid Engine's `qmon` or `qstat` shows the reservation's queue in state `C` for "suspended by calendar."
`Reserved`	Sun Grid Engine's execution host's generic queue's resources have been reduced to those needed for this reservation. This situation ensures that no other Sun Grid Engine job is using these resources when the reservation requires the resources. A reservation should move to this state before the reservation's start time. The amount of time before is equal to the configuration property `AdvanceReservation.MaximumNonreservedJobDuration` (in `defaults.prop`, 1 hour is a suggestion). A nonreservation job should start during that window only if the job has other resources available to it (that is, only if the job is compatible with all reservations).
`Started`	The reservation's Sun Grid Engine queue is active (or the queue's Sun Grid Engine calendar should have made the queue active). This situation should occur at the reservation's start time. Sun Grid Engine's `qmon` or `qstat` shows the reservation's queue in state " " (the space means active).

TABLE 5-2 Reservation States *(Continued)*

Reservation State	Description
`Finished`	The reservation's Sun Grid Engine queue is suspended. (That is, the queue should be inactive or Sun Grid Engine's calendar should have suspended the queue, which includes suspending any jobs still running on the queue). This situation should occur at the reservation's finish time (its duration period after the reservation's start time). Sun Grid Engine's `qmon` or `qstat` shows the reservation's queue in state `C` again (suspended by calendar). You won't see a reservation in this state, because the reservation immediately transitions to `Returned`. A Sun Grid Engine administrator can resume any suspended jobs.
`Returned`	The reservation's resources have been returned to the execution host's generic queue, because the reservation is done. This situation occurs at the reservation's finish time.
`QueueGone`	The reservation's Sun Grid Engine queue has been deleted (along with any jobs pending for this queue and the job's subordinate calendar and user set). A reservation should move to this state considerably after the reservation's finish time, to enable the status of the queue to be evaluated or pending jobs to be moved to another queue, if needed. The amount of time after is equal to the configuration property `AdvanceReservation.FinishToDeleteQueueDuration` (in `defaults.prop`, 24 hours is a suggestion).

Advance Reservation Troubleshooting

General Troubleshooting

Both the client and server contain both assertions and Assume code. The Assume code prints stack traces if the code encounters an error, but then continues to execute (rather than aborting).

Client Troubleshooting

The client displays the following message to indicate the client doesn't find the server running (on the port specified):

```
Cannot connect to server java.net.ConnectException:
Connection refused on port 6789
```

Server Troubleshooting

- The server prints every message received from a client and sent to a client, as well as other messages. These messages are helpful in identifying problem cases.
- When the server starts, the server deletes any reservations from the database that are sufficiently past the reservation stop time (the reservation's start time plus the reservation's duration). Additionally, the reservations are no longer visible in the Reserve GUI.
- The server is multithreaded, but only one thread accepts client connections, so reservations are currently made one at a time.
- Graphics allocated to a host do not appear in `-sq` output. Configure consumable attributes by queue (`all.q@`*myhost*) rather than by host, or edit the `limits` file to reflect the correct attributes, as displayed for any $*hostname* using:

```
% qconf -se $hostname | more +/complex_values -1
```

APPENDIX **A**

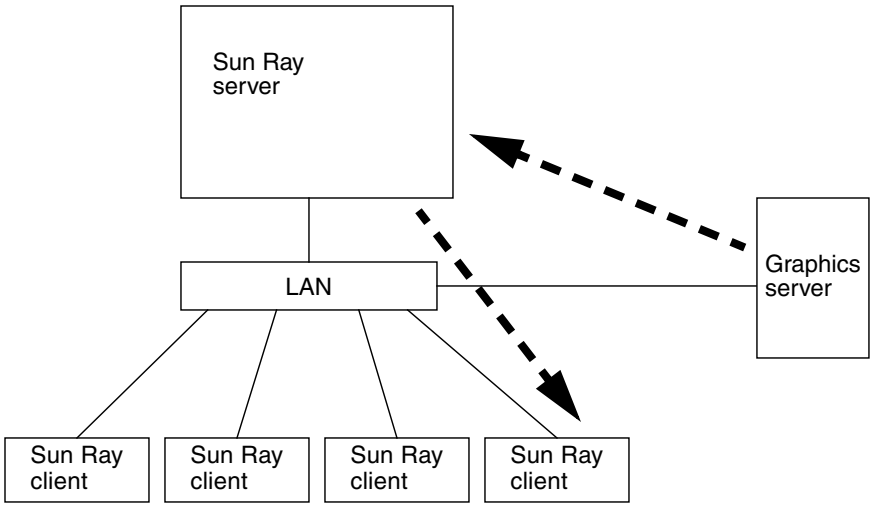
Sun Ray Network Architectures and VirtualGL

This appendix discusses constraints and behaviors between three types of Sun Ray network architectures and VirtualGL. Topics include:

Sun Ray Plug-In for VirtualGL

To display images on a Sun Ray client from another system, that system sends the images to the Sun Ray user's 2D X server on the Sun Ray server. The Sun Ray server then sends the images to the Sun Ray clients. See FIGURE A-1.

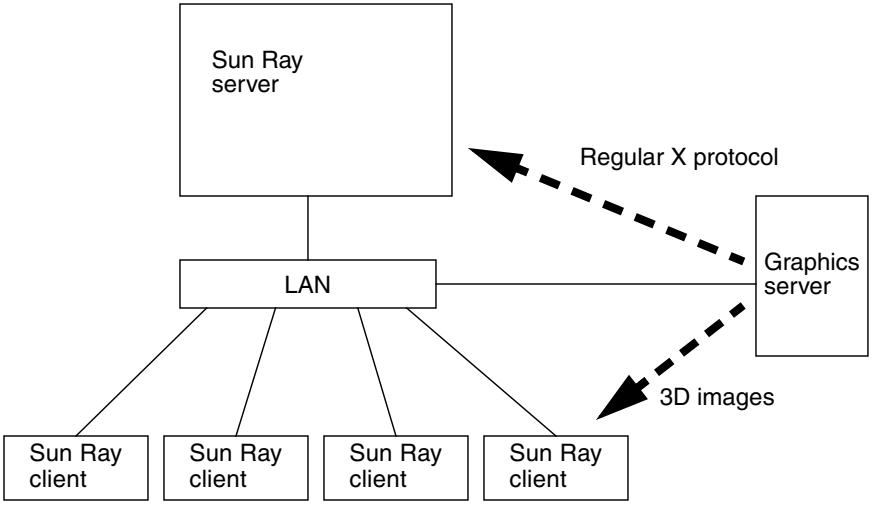
FIGURE A-1 Traditional Graphics Serving



This situation works well for applications that don't send images at a high rate. However, if using VirtualGL, a single user can generate considerable network traffic to sustain an acceptable frame rate. Multiple users requesting similar services could quickly overwhelm the networking capability of a Sun Ray server.

As a possible solution to this problem, the Sun Shared Visualization 1.1 software includes a Sun Ray plug-in for VirtualGL (the `SUNWvglsr` Solaris package or the `VirtualGL-SunRay` RPM for Linux). This plug-in enables VirtualGL to send images directly to the Sun Ray using Sun Ray protocols. See FIGURE A-2.

FIGURE A-2 Sun Ray Plug-in



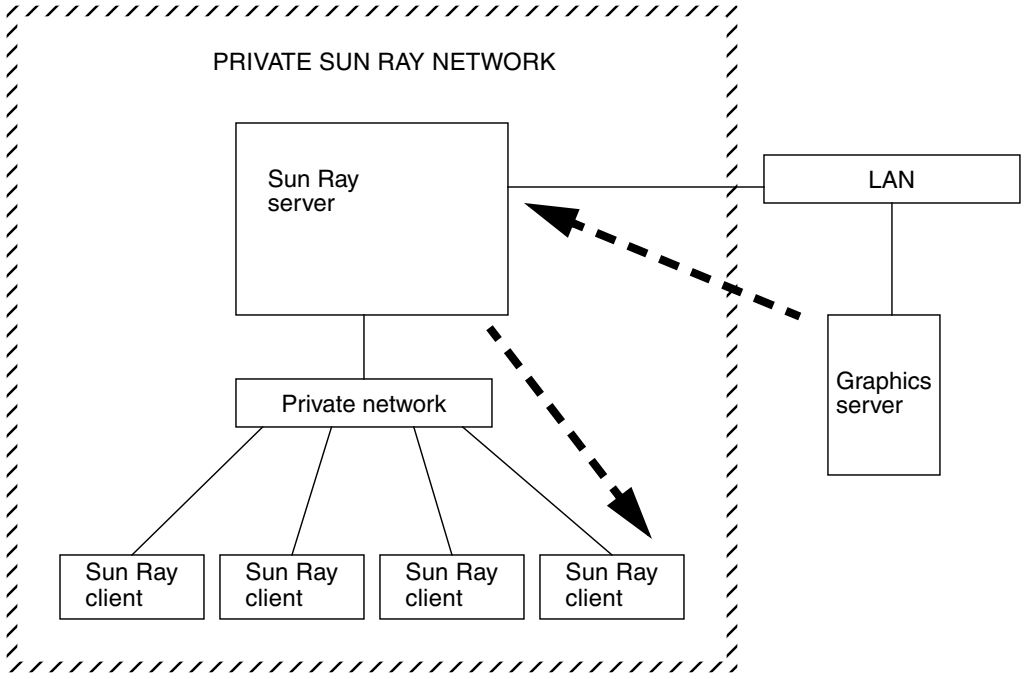
The current default compression method used for the plug-in is quite lossy but can be controlled with options and VGL_ environment variables. See the appendix, "VirtualGL Reference", in the *Sun Shared Visualization 1.1 Software Client Administration Guide*, 820-3257.

The advantage of this architecture is that the majority of the network load is off of the Sun Ray server, making the model more scalable. If the connection between the Sun Shared Visualization 1.1 server and the Sun Ray clients is a network switch, then the model also avoids a network bottleneck.

Private Sun Ray Networks

Some Sun Ray networks are private, in that only the Sun Ray server has access to the Sun Ray clients. See FIGURE A-3.

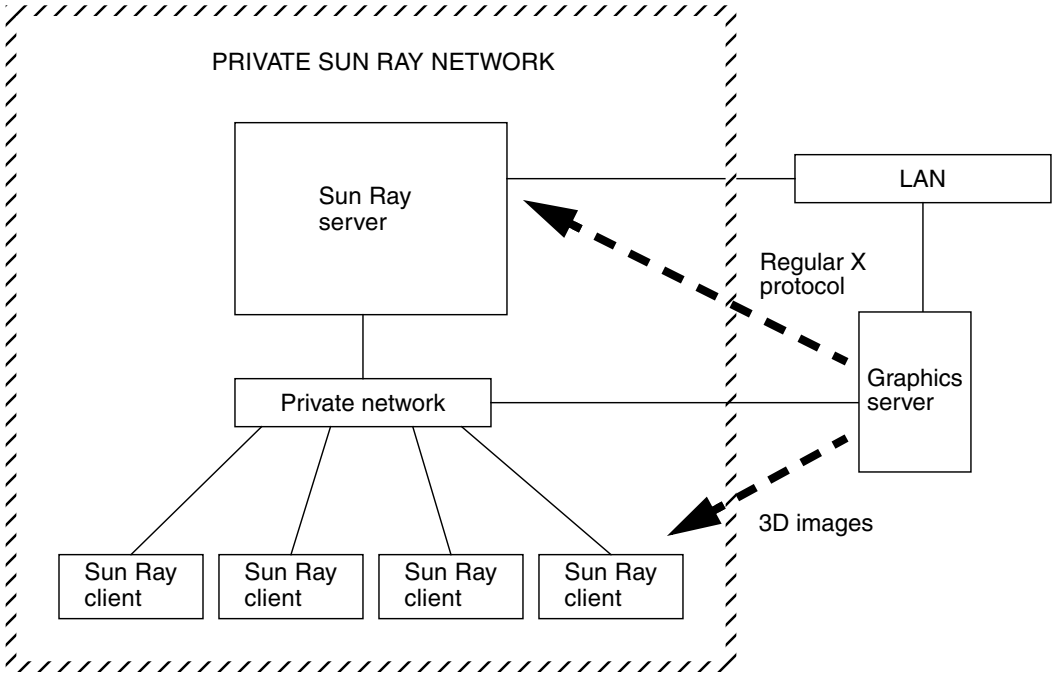
FIGURE A-3 Private Sun Ray Network



In this situation, there is no direct network path from the Sun Shared Visualization 1.1 server to the Sun Ray clients, so standard VirtualGL methods need to be used to transmit the images. VirtualGL needs to use X11 Image Transport, which is requested using the `-c proxy` option on the `vglrun` command line or setting the `VGL_COMPRESS` environment variable to `proxy`.

This configuration might work well for light use, but is not advised for common VirtualGL usage because of the network load that is put upon the Sun Ray server. When practical, an alternative is to use a second Ethernet port on the Sun Shared Visualization 1.1 server to include the server in the private Sun Ray network. See FIGURE A-4.

FIGURE A-4 Semi-Private Sun Ray Network



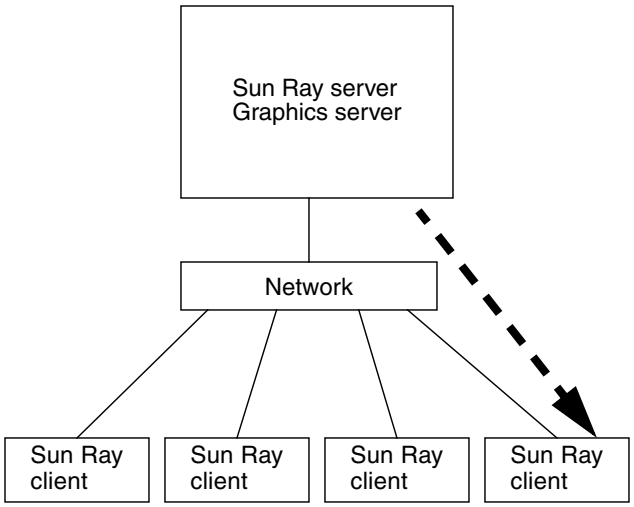
The advantage of this method is that the Sun Ray plug-in enables lower network load on the Sun Ray server, without changing the Sun Ray network architecture. Documentation on how to configure this network architecture is beyond the scope of this document.

An alternative solution is to put the Sun Ray clients onto the LAN. See the *Sun Ray Administration Guide* for instructions on how to do this.

Sun Ray Server as a Shared Visualization 1.1 Server

It is possible for the Sun Ray server and the Sun Shared Visualization 1.1 server to be the same system if there is graphics accelerator hardware on the Sun Ray server. See FIGURE A-5.

FIGURE A-5 Sun Ray Server as a Shared Visualization 1.1 Server



In this case, you may choose between using the Sun Ray plug-in, which is the default, and disabling the Sun Ray plug-in so that VirtualGL uses the X11 Image Transport to give images to the Sun Ray X server, and the Sun Ray server does the compression and transmission of the images to the Sun Ray clients. The latter technique can improve image quality with little or no effect on performance.

The Sun Ray plug-in is disabled by one of these methods:

- Use the `-c proxy` option on the `vglrun` command line.
- Set the `VGL_COMPRESS` environment variable to `proxy`.
- Remove the Sun Ray plug-in software altogether (this is the `SUNWvglsr` Solaris package or the `VirtualGL-SunRay` RPM for Linux).

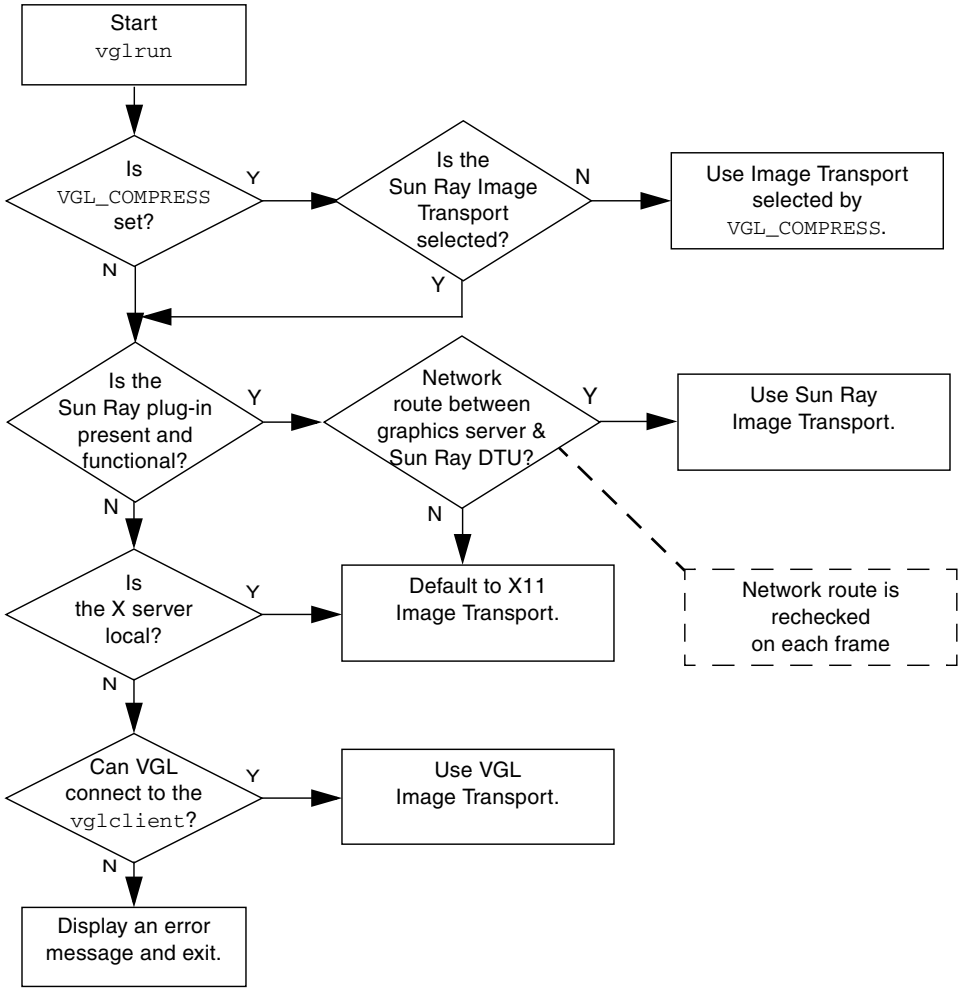
This architecture works best when it is practical to do graphics resource allocation at the time of Sun Ray session login. This happens in situations where all Sun Ray users access 3D applications, or 3D usage is relatively predictable as a function of the number of users.

If, however, the Sun Ray server is hosting a large majority of users who are not using 3D applications, the users who do so consume a disproportionate amount of resources on the network. This situation makes load management on the Sun Ray server more challenging.

VirtualGL Behavior on Sun Ray Networks

The flowchart in FIGURE A-6 describes the behavior of VirtualGL in a Sun Ray network. For more information about `VGL_COMPRESS`, see Appendix A of the *Sun Shared Visualization 1.1 Software Client Administration Guide.*

FIGURE A-6 Behavior of VirtualGL in a Sun Ray Network



APPENDIX **B**

Application Recipes

This appendix provides recipes for using Sun Shared Visualization 1.1 software with selected applications. Topics include:

Recipes for Selected Applications

Note – Environment variables for Sun Grid Engine jobs either must be saved with the job using the `-v` *variable* option of `qsub` or be set by the job (script) after its invocation by Sun Grid Engine.

TABLE B-1 lists some application recipes.

TABLE B-1 Example Application Recipes

Application	Platform	Recipe	Notes

ANSA v12.1.0	Linux/ x86	At the top of the `ansa.sh` script, add the following:
		`LD_PRELOAD_SAVE=$LD_PRELOAD`
		`export LD_PRELOAD=`
		Then add:
		`LD_PRELOAD=$LD_PRELOAD_SAVE`
		just prior to this line:
		`$(ANSA_EXEC_DIR)bin/ansa_linux$( ext2)`
			The ANSA startup script directly invokes `/lib/libc.so.6` to query the `glibc` version. Since the VirtualGL faker depends on `libc`, preloading VirtualGL when directly invoking `libc.so.6` creates an infinite loop. So it is necessary to disable the preloading of VirtualGL in the application script, and then reenable preloading prior to launching the actual application.

TABLE B-1 Example Application Recipes *(Continued)*

Application	Platform	Recipe	Notes
AutoForm v4.0x	All	`vglrun +sync xaf_`*version*	AutoForm relies on mixed X11/OpenGL rendering, and thus certain features (particularly the Dynamic Section dialog and Export Image feature) do not work properly unless `VGL_SYNC` is enabled. Since `VGL_SYNC` automatically enables the X11 Image Transport and disables frame spoiling, ensure that you use TurboVNC when `VGL_SYNC` is enabled.
I-deas Master Series 9, 10, & 11	Solaris SPARC	When running I-deas with VirtualGL on a Solaris SPARC server, remotely displaying to a non SPARC client host or to an X proxy such as VNC, set the `SDRC_SUN_IGNORE_GAMMA` environment variable to `1`.	I-deas aborts if the software detects that the assigned X visual is not gamma-corrected. Gamma-corrected X visuals only exist on Solaris SPARC X servers, so if you are displaying the application using another type of X server or X proxy that doesn't provide gamma-corrected X visuals, then you must override the gamma detection mechanism in I-deas.
Java 5 2D applications that use OpenGL	Solaris SPARC	When VirtualGL is used in conjunction with Java 5 (Java 1.5.0) software to remotely display Java 2D™ applications using the OpenGL pipeline, certain Java 2D applications cause the OpenGL subsystem to crash with the following error: `thread tries to access GL context current to another thread` If you encounter this error, set the `SUN_OGL_IS_MT` environment variable to `1` and rerun the application.	This issue does not exist in Java 6 software.

TABLE B-1 Example Application Recipes *(Continued)*

Application	Platform	Recipe	Notes

Java 2D applications that use OpenGL	Linux, Solaris OS	Java 2D uses OpenGL to perform rendering if `sun.java2d.opengl` is set to `True`. For example:
		`java -Dsun.java2d.opengl=True MyAppClass`
		In order for this to work in VirtualGL, you must start `vglrun` with the `-dl` switch. For example:
		`vglrun -dl java -Dsun.java2d.opengl=True MyAppClass`
		If you are using Java v6 b92 or later, you can also set the environment variable `J2D_ALT_LIBGL_PATH` to the path of `librrfaker.so`. For example:
		`setenv J2D_ALT_LIBGL_PATH /opt/VirtualGL/lib/librrfaker.so`
		`vglrun java -Dsun.java2d.opengl=True MyAppClass`	

Pro/ENGINEER Wildfire v3.0	Solaris SPARC	On releases prior to 1.1.1, when using VGL Image Transport, set the environment variable `VGL_INTERFRAME` to `0` on the graphics server prior to launching Pro/ENGINEER v3.0. This recipe is not needed on Sun Shared Visualization releases beginning with release 1.1.1.	Pro/ENGINEER 3.0 frequently sends long sequences of `glFlush()` calls though nothing new has been rendered. The `glFlush()` calls cause VirtualGL to send long sequences of duplicate images into the VGL Image Transport image pipeline. If interframe comparison is enabled, the overhead of comparing these duplicate images affects Pro/ENGINEER performance when zooming in or out. Better performance is achieved by disabling interframe comparison and enabling VirtualGL's frame spoiling functionality.
Pro/ENGINEER Wildfire v2.0	Solaris SPARC	Add the following to `~/config.pro`: `graphics opengl` You might also need to set the `VGL_XVENDOR` environment variable to "Sun Microsystems, Inc." if you are running Pro/ENGINEER 2.0 over a remote X connection to a Linux or Windows VirtualGL client.	Pro/ENGINEER 2.0 for Solaris OS disables OpenGL if the application detects a remote connection to a non Sun X server.

TABLE B-1 Example Application Recipes *(Continued)*

Application	Platform	Recipe	Notes
QGL (OpenGL Qt Widget)	Linux	`vglrun -dl` *application*	Qt can be built such that Qt either resolves symbols from `libGL` automatically or uses `dlopen()` to manually resolve those symbols from `libGL`. As of Qt v3.3, the latter behavior is the default, so OpenGL programs built with later versions of `libQt` do not work with VirtualGL unless the `-dl` switch is used with `vglrun`.
Wine	Linux	`vglrun -dl wine` [*windows-opengl-app*`.exe`]	Intercept `dlopen()` call for `libGL.so`.

APPENDIX C

Manual Configuration Information

This appendix provides instructions on some manual procedures that are alternatives to the configuration procedures in Chapter 4. If you perform the procedures as described in Chapter 4, which make use of the `vglserver_config` script, you do not need the information in this appendix.

This appendix contains configuration information for both Solaris and Linux based Sun Shared Visualization 1.1 servers.

Adding Graphics to Sun Grid Engine Manually

This section describes how to add graphics resources to Sun Grid Engine. This section is an alternative to the procedures in "Adding Graphics to Sun Grid Engine" on page 64. The procedures in Chapter 4 should be simpler to perform in most cases than the alternative procedures here. However, if you need to set or update graphics resources manually, either procedure can be followed.

There is more information regarding the graphics resources and their meaning, with advice on sizing, in "Sun Grid Engine Graphics Resources" on page 70.

You must first install Sun Grid Engine and the Sun Shared Visualization 1.1 software before continuing with the procedures in this appendix.

These steps are to be performed as the `sgeadmin` user on the queue master host, or on an administrative host that mounts `$SGE_ROOT` read-write.

▼ To Set the Variables

1. **Set the `$SGE_ROOT` and `PATH`:**

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

   where `/gridware` is the base directory of your `$SGE_ROOT`.

2. **Set your `DISPLAY` environment variable to the system whose X server keyboard you are using:**

```
% setenv DISPLAY myhost:0.0
```

   where *myhost* is the hostname of the X server and `:0.0` identifies the X screen and display.

▼ To Add Graphics to Sun Grid Engine

Note – If you are upgrading an existing Sun Shared Visualization software installation, you only need to perform Step 1, Step 8, and Step 9.

---

1. **If the optional software was not already installed on the grid's NFS server, then, as superuser, install that software.**
   - On a Solaris NFS server, install the `SUNWsge3D` package into the `$SGE_ROOT` directory:

```
# pkgadd -d download-directory SUNWsge3D
```

Note – Ensure that your `$SGE_ROOT` value is your answer to the installation prompt, "`Please enter your SGE_ROOT directory.`"

   - On a Linux NFS server, install the `sun-n1ge-3D.noarch.rpm` package into the `$SGE_ROOT` directory:

```
# rpm -iv /path-to-rpm-file/sun-n1ge-3D.noarch.rpm
```

2. **Set an administrative email for Sun Grid Engine so all errors are reported by email.**

a. **Type:**

```
% qconf -mconf
```

This command starts your `$EDITOR` with a file containing configuration variables.

b. **Add the email address for the `administrator_mail` configuration variable, and save and quit the file.**

3. **Add resource names to the Sun Grid Engine complex.**

The complex is the vocabulary of variables that can be specified. The most important resources to add are described in TABLE C-1 and explained in "More Graphics Resource Allocation Information" on page 74.

TABLE C-1 Resources to Add to Sun Grid Engine Complex

Resource Name	graphics	maximum_graphics	graphics_alone	chromium
Shortcut	gfx	maxgfx	alone	cr
Type	INT	INT	INT	INT
Relation	<=	<=	<=	<=
Requestable	YES	YES	YES	YES
Consumable	YES	NO	NO	YES
Default	0	0	0	0
Urgency	0	0	0	0

Use one of the following three ways to add resource names:

- Use the `add_to_complex` script, as described in "To Add Graphics to Sun Grid Engine" on page 65. The script adds the information in TABLE C-1 to your Sun Grid Engine complex.

  In addition, the script adds resources for use with Sun Scalable Visualization software. These resources are named `dedicated_graphics`, `headnode`, `sc_rows`, and `sc_cols`. This option is the only way to add resource names if interaction with Sun Scalable Visualization software is necessary.

- Use the graphical tool, `qmon`.

a. **Start `qmon`.**

b. **Click on Complex Configuration.**

The complex is displayed.

c. **Add new entries to define new resources, using the information in TABLE C-1.**

d. Click Add to add each new resource and Commit to save the updated complex.

- Use `qconf`'s modify complex command.

a. Start `qconf -mc`.

This command opens your `$EDITOR` with the complex configuration.

b. Copy the `slots` line, and edit subsequent copies to look like the following:

```
graphics          gfx     INT     <=      YES     YES     0 0
maximum_graphics  maxgfx  INT     <=      YES     NO      0 0
graphics_alone    alone   INT     <=      YES     NO      0 0
chromium          cr      INT     <=      YES     YES     0 0
```

c. Save and quit the file.

d. Verify the complex configuration with `qconf -sc`.

The output can be minimized with just headings and the graphics lines:

```
% qconf -sc | sed -n '1p;/graphics/p'
```

4. Define which hosts have how many graphics resources available.

This is the maximum number of simultaneous graphics jobs that Sun Grid Engine could start on that host. For example, if your host has two graphics boards and the boards can accommodate three jobs each, your resources would be 2 x 3, or 6.

There are two ways to define graphics resources:

- Using `qmon`

a. Start `qmon`.

b. Click the Host Configuration button.

c. Click the Execution Host tab.

d. Select the host to specify the graphics resource.

e. Click the Modify button.

f. Click the Consumables/Fixed Attributes tab.

g. Click the Name button, select the resource name (for example, graphics), and click OK.

h. In the Value field, type the number of graphics cards available on this host.

i. Repeat Step g and Step h for the other resource names (`maximum_graphics`, `graphics_alone`, `chromium`, and so on).

j. Click OK and click Done.

Note – For more information about graphics resources, see "More Graphics Resource Allocation Information" on page 74.

---

k. Repeat from Step d for every host with graphics resources shared through Sun Grid Engine.

- Using the command line:

a. Type the `qconf` command:

```
% qconf -mattr exechost complex_values resourcename=value hostname
```

where *resourcename* is the names provided in TABLE C-1 and *value* is the number of resources on the *hostname*.

Note – Set the value of `graphics_alone=1` to enable a dedicated graphics accelerator. Set the value of `chromium=1` to identify the host as a Chromium head node.

---

b. Verify the setting:

```
% qconf -se hostname
```

The output is a list similar to:

```
qconf -se hostname
hostname              hostname
load_scaling          NONE
complex_values        graphics=2
:
:
```

Piping the output through `grep` can list just the complex values:

```
% qconf -se hostname | grep graphics
```

- Alternatively, you can allocate graphics resources to queues, instead of execution hosts. This action is beneficial when using the Advance Reservation facility.

a. Use the `qconf` command to set graphics resources to queues:

```
% qconf -mattr queue complex_values graphics=value queue@hostname
```

b. Verify the `complex_values` file by typing one of the following commands:

```
% qconf -sq queue | grep graphics
```

```
% qconf -sq queue@hostname | grep graphics
```

5. Set the starter and epilog scripts for Sun Grid Engine's `all.q` cluster queue.

These scripts are hooks supported by Sun Grid Engine to provide queue-specific activity before and after a job runs. There are several ways to set these scripts

a. As an SGE administrative login or as `root`, type:

```
% cd $SGE_ROOT/graphics
% ./use_standard
```

This action sets the starter and epilog scripts for all Sun Grid Engine queues.

b. Use the `qconf` command to set the `starter` script:

```
% qconf -mattr queue starter_method $SGE_ROOT/graphics/starter all.q
```

c. Use the `qconf` command to set the `epilog` script:

```
% qconf -mattr queue epilog $SGE_ROOT/graphics/epilog all.q
```

d. Use the `qconf` command to edit the queue:

```
% qconf -mq all.q
```

e. Use `qmon`'s Queue Control panel.

i. Type:

```
% qmon
```

ii. Select the Cluster Queue `all.q`**.**

iii. Click the Modify button at the right.

iv. Find the Execution Method tab.

v. In the fields for Epilog and Start Methods, type the path. For example:

```
Epilog                  /gridware/sge/graphics/epilog
Starter Method          /gridware/sge/graphics/starter
```

6. **(Optional) Copy the** `graphics/docs/README` **file to a more user accessible location.**

Tip – The contents of the `README` file summarize Sun Grid Engine use. Edit the file to better describe your particular site, rename the file, and make the file available to users in `$SGE_ROOT`.

---

7. **Ensure that your** `DISPLAY` **environment variable refers to your X server:**

```
% setenv DISPLAY myhost:0.0
```

where *myhost* is the hostname of the X server.

8. **Attempt to run a graphics job.**

This example submits to any Shared Visualization graphics server:

```
% qrsh -b n /opt/VirtualGL/bin/vglrun -c proxy -spoil \
$SGE_ROOT/graphics/RUN.glxspheres
```

where:

- `-b n` means the `vglrun` script is a Sun Grid Engine job script with options for your Sun Grid Engine job.
- `-c proxy` enables proxy mode so `vglclient` is not needed (however, performance will be reduced).
- `-spoil` disables frame spoiling, slowing rendering to display speed.

Note – This step uses the `-c proxy` option, which usually is not recommended due to its impact on performance. However, using this option here simplifies the verification process without ongoing impact.

The next example names a graphics execution host:

```
% qrsh -b n -q all.q@hostname /opt/VirtualGL/bin/vglrun -c proxy -spoil  \
$SGE_ROOT/graphics/RUN.glxspheres
```

9. **Start** `$SGE_ROOT/graphics/empty_jobs` **from** `/etc/init.d/sgeexecd`**.**

   `/etc/init.d/sgeexecd` is the Sun Grid Engine standard startup script and the script initiates shepherd processes. If these processes are shut down before the graphics jobs, you cannot reclaim the resources of those graphics jobs. To alleviate any possibility of this problem:

   a. **Edit the** `/etc/init.d/sgeexecd` **file and around line 245, find** `$bin_dir/sge_execd`**.**

   b. **Insert the following text before that line:**

   ```
   pgrep -u sgeadmin sge_execd || $SGE_ROOT/graphics/empty_jobs
   ```

   Replace `sgeadmin` if your site uses a different SGE administrative login.

Note – Unless `/etc/init.d/sgeexecd softstop` was used, graphics jobs that are still running when `execd` is shut down lose their the `sge_shep` shepherd processes, so the epilog script is not started for the jobs. Consequently, the job allocator does not know about any graphics resources being consumed by such orphan jobs.

Note – You need to repeat this step if the Sun Grid Engine software is upgraded.

Index

T

U

V

X