Deep Computing Visualization

# Installation and User Guide

*Version 1 Release 4*

G224-7622-00

**Note:**
Before using this information and the product it supports, read the information in "Notices" on page 69.

**Fourth edition (October 2007)**

This edition replaces G224-9183-02 for program number 5724-K69.

IBM welcomes your comments. A form for readers' comments is provided at the back of this publication, or you can address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America
FAX: +1-845-432-9405
IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com

If you would like a reply, be sure to include your name, address, and telephone number.

Make sure to include the following information in your comment or note:
- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

# Contents

# Figures

# Tables

# About this document

This document describes the installation, configuration, and operation of Deep Computing Visualization.

The document describes how to use the software to:

- Enhance graphical rendering of complex OpenGL applications
- Provide scalable visualization of OpenGL applications using a distributed rendering cluster
- Provide high-performance visualization across low-bandwidth, high-latency networks to remote or distributed locations

**Save this document**

Retain this document with your original system. This document emphasizes recent system information and does not include complete information about previous releases.

## Who should use this document

This document is designed for:

- Users of OpenGL applications
- System administrators
- Developers who intend to extend the software using SVN Overload capabilities.

## Typographic conventions

This document uses the following typographic conventions:

- Commands appear in bold font when they appear in text.
  - Example: **logon**
- File and directory names are italicized when they appear in text.
  - Example: */etc/passwd*
- Variables for user-supplied information on commands are indicated by italics.
  - Example: **telnet** *app_host*
- Literals in commands appear in monospace font.
  - Example: `rpm -i --force` *`abc.rpm`*
- Angle brackets (< >) indicate that you must make a substitution.
  - Example: `<username>` indicates that you must supply your user name.
- Square brackets ([ ]) indicate that the enclosed values are optional.
- Curly brackets ({ }) indicate that you must select one of the enclosed values.
- Multiple options for a value are separated by a vertical bar ( | ).
- Default values are underlined.

## Related information

For the latest information about Deep Computing Visualization, see the documents at http://www.ibm.com/servers/deepcomputing/visualization.

## Accessibility information

Accessibility information for IBM products is available online at the IBM Accessibility Center. Go to http://www.ibm.com/able, then click the link for *Product accessibility information.*

# What's new

## Deep Computing Visualization 1.4

Changes in this edition include:

- **New information:**
  - Startup procedures on Linux®
  - SVN on Microsoft® Windows® XP (32-bit and 64-bit)
  - Additional support for OpenGL and GLX
  - Support for TCP/IP, SDP, and OpenMPI
  - SVN Listener
  - New error messages
  - NVidia framelocking support
  - References to multiple applications running under RVN
  - Restructured document layout
- **Updated information:**
  - The `rvn_enable` command on Linux supersedes the `rvn_sender` command
  - The `svn_enable` command on Linux supersedes the `svn_sender` command
  - Streamlined startup process for RVN and SVN on Linux
  - SVN can now run 32-bit applications on 64-bit systems
  - RVN can now run 32-bit applications on 64-bit Windows systems
  - RVN Dashboard and Launcher concepts and procedures
  - Revised installation procedures
- **Deleted information:**
  - References to the **`svn_sender`**, **`rvn_sender`** and **`rvn_sender.bat`** commands
  - Obsolete error messages
  - Support for Scali, Topspin
  - Removed several environment variables
  - Removed RVN sessions information

## Deep Computing Visualization 1.3

Changes in this edition include:

- **New information:**
  - Added information about RealVNC Visualization Edition
    - Use of this version of RealVNC improves the usability of the interface to DCV. For example, it eliminates the need to specify an additional conference ID and conference key when joining an RVN session.
    - RealVNC Visualization Edition is now the only supported VNC software (other VNC products — from RealVNC or other companies — are not supported).
  - Added support for RVN on application hosts on 32-bit Windows XP
    - RealVNC Visualization Edition must be used for RVN sessions with a Windows application host. X11 Export mode is not supported.

- The new  command can be used from these application hosts, similar to the `rvn_sender` command for Linux application hosts.

– Added information about the RVN Launcher, a graphical user interface for Linux and Windows users that can be used instead of the `rvn_sender` or rvn_sender.bat command.

– Added information about SVN support for DMX servers.

– Added information about the `rvn_viewer` command, which is used instead of the `vncviewer` command by Linux end stations to start the VNC Viewer.

- **Updated information:**
  – Revised the information about X11 Export mode for RVN sessions to clarify that it can be used only for application hosts running under Linux.
  – Updated the description of the RVN Dashboard.
  – SVN supports OpenGL up through specification level 2.0, including support for shader programming.
  – Changed references to the SSC component of RVN to *accelerated graphics*.
  – Corrected the name of the `SVN_SYNC_ON_RETRACE` environment variable. The correct name is `SVN_SWAP_ON_RETRACE`.
  – Moved information about environment variables to an appendix.
- **Deleted information:**
  – Removed the requirement for the pthreads compatibility library when running RVN under Windows.
  – Removed information about Microsoft Windows 2000 as a platform for RVN end stations.
  – Removed the `RVN_DASHBOARD_DELAY` and `RVN_DASHBOARD_PERMIT_CLOSE` environment variables.

## Deep Computing Visualization 1.2

Changes in this edition include:

- **New information:**
  – Added RVN end station support for xSeries® rendering servers using Microsoft Windows 2000 or Microsoft Windows XP
  – Added support for simultaneous SVN and RVN sessions
  – Added support for RVN rendering servers to dynamically join and leave a session
  – Added the ability to override OpenGL functions with user-implemented modules
- **Updated information:**
  – Revised SVN commands
  – Revised RVN commands
  – Revised installation procedures
- **Deleted information:**
  – Removed `-any` and `-loop` options for the `rvn_receiver` command
  – Removed `-n` option and `endstation` list for the `rvn_sender` command
  – Removed `RVN_LISTEN_ANY` and `RVN_LISTEN` environment variables

# Chapter 1. Introducing Deep Computing Visualization

Deep Computing Visualization provides a scalable and collaborative middleware infrastructure to support and enhance the graphics functions of OpenGL applications on Linux and Microsoft Windows XP.

Deep Computing Visualization can display complex visual data:

- On large tiled displays capable of displaying very high resolution images
- Across multiple, simultaneous distributed displays using low-bandwidth networks.

## Overview

In a typical visualization scenario, a software application sends a stream of graphics commands to a graphics adapter through an input/output (I/O) interface. The graphics adapter renders the data into pixels (stored as raster content in the video memory of the adapter) and outputs them to the local display as a video signal.

Deep Computing Visualization uses high-speed local area networks (LANs) to link a rendering cluster to an ultra-high resolution display, and low-speed wide area networks (WANs) to provide efficient and secure transport of image data to remote displays.

In both cases, the software inserts an *intercept* OpenGL library into the graphics software stack of a host system running a 3-D graphics application.

- In LAN scenarios, the scene geometry and graphics state are transmitted to a cluster of rendering servers. This is referred to as *Scalable Visual Networking*.
- In WAN scenarios, the scene geometry and graphics state are rendered locally, and pixels are sent to remote displays. This is called *Remote Visual Networking*.

## Scalable Visual Networking

Scalable Visual Networking (SVN) converts an OpenGL display running on a single application host into a multi-tile display driven by a cluster of rendering servers.

An SVN-enabled application virtualizes the rendering pipeline over a high-speed LAN. An SVN intercept OpenGL library is installed on the application host running the 3-D visualization application. The library accepts the OpenGL calls made by the application, encodes them, and broadcasts the call data to a cluster of rendering servers over a high-speed, low-latency network. Each rendering server in the cluster receives the encoded data stream, decodes it, and calls the corresponding OpenGL command.

Each server in the cluster can render and display its portion of the final image as a tile on a display wall or projection system.

## Remote Visual Networking

Remote Visual Networking (RVN) provides remote, collaborative access to OpenGL graphics applications.

RVN intercepts data within the rendering pipeline, uses local resources for rendering, and sends the finished images to remote displays (*end stations*). It permits the use of 3-D applications even in high-latency and low-bandwidth networks. Because it uses local rendering resources, RVN provides end stations with the ability to display complex interactive 3-D applications. RVN can also send images to multiple destinations, providing simultaneous interaction by multiple collaborators.

For more information, see:


- Chapter 6, “Using Scalable Visual Networking,” on page 21
- Chapter 7, “Using Remote Visual Networking,” on page 27
- Chapter 8, “SVN and RVN interoperability,” on page 39

# Chapter 2. Prerequisites for installation

This chapter lists the hardware and software prerequisites for installing SVN and RVN.

## SVN prerequisites

SVN requires the following hardware systems:

- An application host to run your graphics application
- A cluster of systems to act as rendering servers, each of which is connected to one or more output displays
- A fast interconnect (LAN) between the application host and the rendering servers

Table 1 lists the hardware requirements for SVN.

*Table 1. Hardware requirements for SVN*

| Device | Requirement |
|---|---|
| Application host | One of the following systems:<br>• IBM IntelliStation Z Pro workstation<br>• IBM System x 3755 |
| Rendering servers | IBM IntelliStation Z Pro workstation |
| Network switch | Cisco/TopSpin InfiniBand network interconnect |
| Graphics card | NVIDIA Quadro FX family of ultra-high-end graphics adapter or NVIDIA Quadro Plex 1000 Models I, II, and IV |

Table 2 lists the software requirements for SVN.

*Table 2. Software requirements for SVN*

| Software | Level |
|---|---|
| Operating system | Red Hat Enterprise Linux 3.0, Update 9<br>Red Hat Enterprise Linux 4.0, Update 5<br>Microsoft Windows XP Professional Service Pack (SP) 2 (32-bit and 64-bit) |
| NVIDIA driver<br><br>Modify */etc/X11/xorg.conf* to associate adapter ports with X Windows servers. | • **Linux:** 100.14<br>• **Windows:** 162.18<br><br>To download the latest NVIDIA graphic adapter drivers, visit http://www.ibm.com/products and use the links for Support and downloads to search for updates. |

## 32-bit and 64-bit operating systems

The application host and rendering servers must have the same version of Linux or Windows installed (all 32-bit or all 64-bit).

**Notes:**

1. SVN supports 32-bit applications running on 64-bit Linux and Windows.
2. If you are using MPI Transport on 64-bit Linux systems, ensure that both 32-bit and 64-bit MPI libraries are installed.
3. If installing on a 64-bit Linux system, you must install the 32-bit RPM of Deep Computing Visualization before installing the 64-bit RPM.

## Optional components

The following components may be required, depending on your system configuration.

### OpenFabrics

OpenFabrics is the recommended driver for Host Channel Adaptor (HCA) hardware, and is typically used for InfiniBand networks. For more information, see www.openfabrics.org.

### MPI

Your deployment might use Message Passing Interface (MPI), a communications protocol for parallel computation. The only supported implementation is OpenMPI. For more information, see http://www.open-mpi.org.

### DMX

Distributed Multiheaded X (DMX) provides multi-head support for multiple displays attached to different systems (each of which is running an X Server). For more information, see http://dmx.sourceforge.net.

## RVN prerequisites

RVN requires the following hardware systems:

- an application host to run an OpenGL application
- one or more end stations, each of them connected to an output display device
- a network connection (WAN) between the application host and the end station

Table 3 lists the hardware requirements for RVN.

*Table 3. Hardware requirements for RVN*

| Device | Requirement |
|---|---|
| Application host | One of the following systems:<br>• IBM IntelliStation Z Pro workstation<br>• IBM System x 3755 |
| End station | Any system (including laptop computers) that can run one of the supported operating systems |
| Graphics card (application host only) | NVIDIA Quadro FX family of ultra high-end graphics adapters or NVIDIA Quadro Plex 1000 Models I, II, and IV |

Table 4 lists the software requirements for RVN.

*Table 4. Software requirements for RVN*

| Software | Level |
|---|---|
| Operating system | Red Hat Enterprise Linux 3.0, Update 9<br>Red Hat Enterprise Linux 4.0, Update 5<br>Microsoft Windows XP Professional Service Pack (SP) 2 (32-bit and 64-bit) |
| Virtual Network Computing (VNC)<br>• VNC Server on the application host<br>• VNC Viewer on the end stations | RealVNC Visualization Edition (RealVNC VE)<br>**Note:** RealVNC VE is the only VNC software that is supported. |
| NVIDIA video driver (application host only)<br>Modify */etc/X11/XF86Config* to associate adapter ports with X Windows servers. | • **Linux:** 100.14<br>• **Windows:** 162.18<br>To download the latest NVIDIA graphic adapter drivers, go to the IBM support site at http://www.ibm.com/products and use the links for Support and downloads to search for updates. |

## 32-bit and 64-bit operating systems

RVN supports application hosts and end stations running 32-bit and 64-bit versions of Linux and Windows. The application host and end stations do not need to run the same operating system to participate in an RVN session.

## RealVNC Visualization Edition

In an RVN session, the application user interface is transferred to an end station using RealVNC Visualization Edition (RealVNC VE).

RealVNC VE is a special edition of RealVNC Enterprise Edition designed exclusively for Deep Computing Visualization. The use of other VNC products (whether from RealVNC or another vendor) is not supported.

For more information about RealVNC VE, go to http://www.realvnc.com/products/visualization.

## Firewall configuration

Both SVN and RVN have similar firewall requirements. Your firewall should allow traffic on the following default TCP ports:

**2007** SVN Listener

**7200-7220**
RVN coordinator (used with X11 Export on Linux)

**7300-7399**
RVN mini-coordinator (used with VNC)

**34567+**
The first of a range of ports used by the SVN cluster of application host and rendering servers. Additional ports are required for each rendering server and are usually acquired sequentially.

**Notes:**

1. These ports can be changed through UI components or environment variables.
2. For SVN, it is recommended that all machines in the cluster are on a trusted network with no firewall restrictions.
3. When the Long Distance Transport option is selected, port 7400 is used for User Datagram Protocol (UDP) data transfer — sequential ports are tried, if necessary, until a successful binding occurs. For more information, see "The RVN mini-coordinator" on page 19.

# Chapter 3. Installing Deep Computing Visualization

See the following sections for OS-specific installation instructions:



**Notes:**

1. Different releases of Deep Computing Visualization on the same system will conflict with each other — uninstall any earlier release before installing this release.
2. For RVN, end stations with release 1.3 are compatible with release 1.4 application hosts.
3. On 64-bit Linux systems, 32-bit versions of the software must be installed before 64-bit versions.
4. On 64-bit Windows systems, both 32-bit and 64-bit versions of the software are installed.
5. The Windows installer also installs the Microsoft Visual C++ 2005 Redistributable and the Intel IPP Integrated Performance Primitives Runtime Installer. Uninstalling either of these will prevent Deep Computing Visualization from working.
6. SVN and RVN are enabled on completion of the installation process. If you update your graphics drivers later, run the **/usr/sbin/dcv_enable** command (as *superuser*) to re-enable the software. (Linux only.)
7. For the latest information about Deep Computing Visualization, including FAQs, go to http://www.ibm.com/servers/deepcomputing/visualization.

## Installing on Linux application hosts and rendering servers

To install on an application host or rendering server, perform the following steps:

1. If you are migrating from an earlier release, back up your wall configuration file and then run the **rpm -e** command to uninstall the previous release.
2. Copy the following RPMs to a temporary directory:
   - dcv.license_acceptance-1.4.i686.rpm
     - This RPM is required the first time you install Deep Computing Visualization.
   - dcv-1.4.0-0.i686.rpm
   - dcv64-1.4.0-0.x86_64.rpm (if you require a 64-bit installation)

   **Note:** If you are using the product CD, mount the CD media in a suitable drive.
3. Log on as the root user.
4. Change the directory (**cd**) to the directory containing the RPMs.
5. Run the following command:

   ```
   rpm –i dcv.license_acceptance-1.4.i686.rpm dcv-1.4.0-0.i686.rpm
   ```

   To continue the installation, you are prompted to accept the license agreement.
6. For a 64-bit installation, you must also run the following command:

```
rpm –i dcv64-1.4.0-0.x86_64.rpm
```

7. For RVN, you also need to install RealVNC VE.
8. To complete the installation, see "Verifying Installation" on page 9.

**Notes:**

1. Deep Computing Visualization is automatically enabled during the installation process. (This result is equivalent to running the `dcv_enable` command on Linux.)
2. On successful completion:
   - The operational files are installed in the */opt/IBM/dcv* directory.
   - The license agreements are installed in the */etc/opt/IBM/dcv/license* directory.
3. The IBM Java runtime environment (JRE) is installed along with the code for the RVN application host.

## Installing on Windows application hosts and rendering servers

To install on an application host or rendering server, log on as an administrator and perform the following steps:

1. Uninstall any earlier release of Deep Computing Visualization.
2. Navigate to the folder containing the installation files.
3. Run the **DCV-setup-1_4.exe** installer and follow the standard instructions for installing Windows applications.

   **Note:** You are prompted to install the Microsoft Visual C++ 2005 Redistributable and the Intel IPP Integrated Performance Primitives Runtime Installer. You must accept these installations. The Visual C++ runtime libraries and Intel installer are isolated from similar installations and will have no effect on other applications.

   The software is installed in the *<Program Files>\IBM\IBM Deep Computing Visualization* directory.
4. Log off and log on again. This action updates the system path so that you can begin using the software.
5. For RVN, you also need to install RealVNC VE.
6. To complete the installation, see "Verifying Installation" on page 9.

The installer modifies the system PATH environment variable to make Deep Computing Visualization files available to RealVNC VE. It will also associate *.svn* and *.rvn* file extensions with the SVN and RVN launchers, respectively, so that you can launch applications from saved profiles.

## Installing on Linux end stations

To install on a Linux system that will be used as an RVN end station, perform the following steps:

1. If you are migrating from an earlier release, run the **rpm -e** command to uninstall the previous release.
2. Copy the following RPM to a temporary directory:
   - dcv.endstation-1.4.0-0.i686.rpm

   **Note:** If you are using the product CD, mount the CD media in a suitable drive.
3. Log on as the root user.

4. Change the directory (**cd**) to the directory containing the RPM.
5. Run the following command:

   ```
   rpm -i dcv.endstation-1.4.0-0.i686.rpm
   ```

6. For RVN, you also need to install RealVNC VE.
7. To complete the installation, see "Verifying Installation."

## Installing on Windows end stations

To install on a Windows system that will be used as an RVN end station, log on as an administrator and perform the following steps:

1. Uninstall any earlier version of Deep Computing Visualization.
2. Navigate to the folder containing the installation files.
3. Run the **DCV-endstation-setup-1_4.exe** installer and follow the instructions for installing Windows applications.
4. For RVN, you also need to install RealVNC VE.
5. To complete the installation, see "Verifying Installation."

**Note:** The installer modifies the system PATH environment variable to make Deep Computing Visualization files available to RealVNC VE.

## Verifying Installation

When the installation procedure is complete, you can verify that SVN and RVN have been properly installed.

### Verification on Linux

To verify SVN installation on Linux, follow these steps:

1. Log on to the application host.

   **Note:** Do not log on as *root* during installation verification.
2. Run the **svn_enable** command, using appropriate *-wall* and *-transport* command-line options. (For best performance, use SOCKET as the *-transport* option.) See "The svn_enable command" on page 21 for additional information.
3. Run an OpenGL application. If the 3-D graphics from the application are displayed in a new window on the local system, then the installation was successful.

To verify RVN installation on Linux, follow these steps:

1. Run the **rvn_enable** command on the application host. For more information, see "The rvn_enable command" on page 33.
2. Run an OpenGL application on the application host — for example, an X11 OpenGL screen saver.
3. Run the **rvn_dashboard** command and wait for the application tab. Select and then clear the **Show on Server** check box to enable and disable the display on the application host.
4. Connect from an end station (on which Deep Computing Visualization has been installed) and verify that the 3-D graphics from the application are visible. If the 3-D graphics from the application are visible on both the end station and the application host, then the installation was successful.

## Verification on Windows

To verify SVN installation on Windows, follow these steps:

1. Ensure that the SVN Listener is running. For more information, see "The SVN Listener" on page 25.
2. Run the SVN Launcher: Click S**tart > All Programs > IBM > IBM Deep Computing Visualization > SVN Launcher**.
3. Select an OpenGL application.
4. Click **Launch**. If the application starts and a new window appears showing the 3-D graphics from the application, then the installation was successful.

**Note:** Some OpenGL application might not work correctly unless you specify their OpenGL DLL Destination directory. For more information, see "The SVN Launcher" on page 23.

To verify RVN installation on Windows, follow these steps:

1. Start the RVN Launcher: Click S**tart >All Programs > IBM > IBM Deep Computing Visualization > RVN Launcher**.
2. Select an OpenGL application.
3. Click **Launch**. The application should start and a new tab should appear on the Launcher (this new tab becomes a dashboard for the specified application).
4. Select and clear the **Show on Server** check box on the dashboard to enable and disable the display on the application host.
5. Connect to the application host from an end station on which Deep Computing Visualization and RealVNC VE have been installed. If the 3-D graphics from the application are visible on both the end station and the application host, then the installation was successful.

**Note:** Some OpenGL applications might not work correctly unless you specify their OpenGL DLL Destination directory. For more information, see "The RVN Launcher" on page 27.

# Chapter 4. Configuring SVN

Before using SVN, you need to configure the application host and rendering servers. Figure 1 shows an example of SVN deployment.

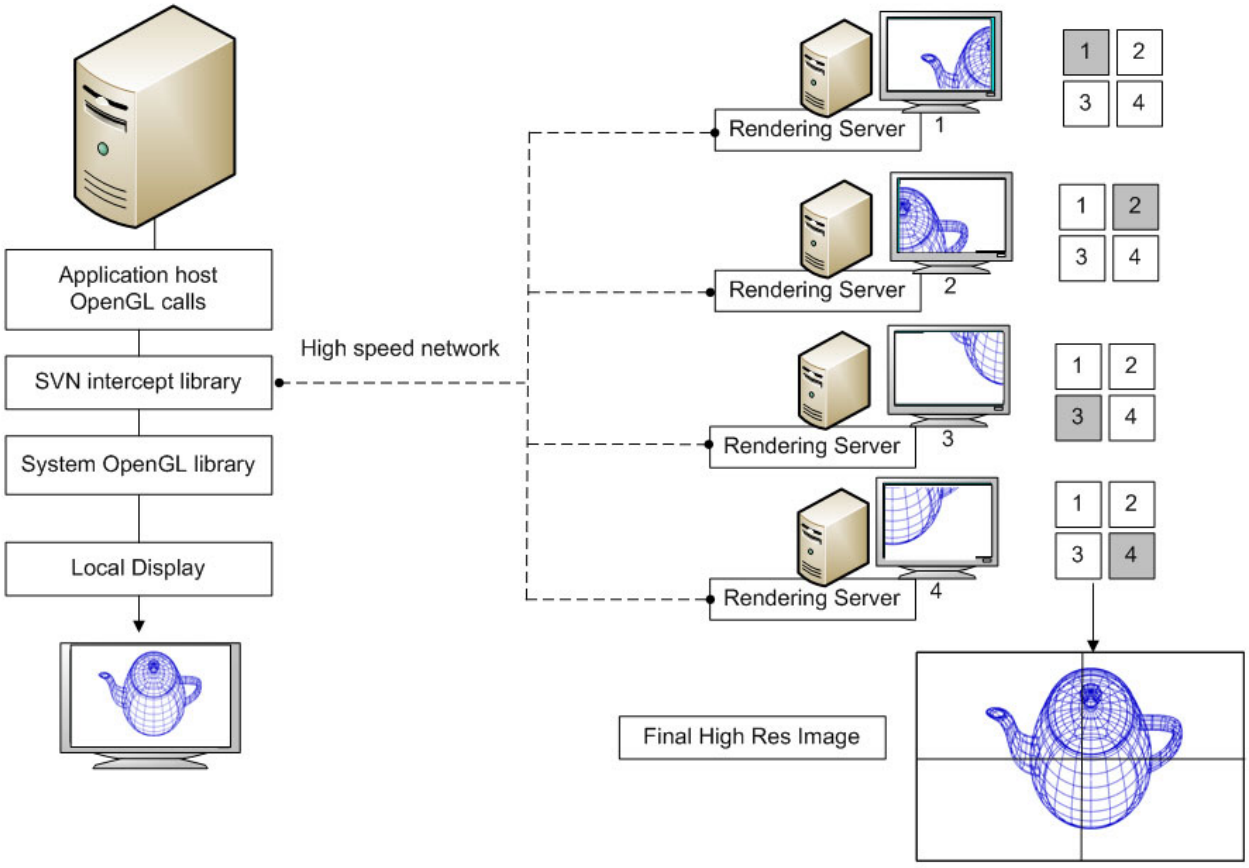



*Figure 1. Deployment of Scalable Visual Networking*

## Display wall configuration

*Tiling* is the term used to describe the configuration of a display wall. This description is defined in a *wall configuration* file that contains an entry for each rendering server. The file must be saved with the file extension "*.cfg*".

**Format of an entry for one rendering server:**

```
<server_name[:d.s]> vwidth vheight x_offset y_offset width height
```

**Syntax of an entry for one rendering server:**

- `server_name[:d.s]` is the X display name for the rendering server.

  **Note:** All system names should be resolvable to IP addresses.
- `d` is the server instance and `s` is the screen number. (Linux only.)
  - If `:d.s` is not supplied, the default is `:0.0`.

- `vwidth` and `vheight` are the overall width and height of the entire multi-tiled display surface.
- `x_offset` and `y_offset` are the starting point of the lower-left corner of the display area (tile) for this server within the entire display surface.
- `width` and `height` are the total width and height for the tile assigned to this server.

**Notes:**

1. You must create a wall configuration file (or use the sample file) before running an application with SVN. For more information, see "Sample wall configuration file" on page 13.
2. The wall configuration file must be stored in a location that is accessible from the application host and readable by non-root or non-administrator users.
3. You must specify the path to the wall configuration file:
   - **Linux**: Specify the path by running the **svn_enable** command or specifying the related environment variable. For more information, see "The svn_enable command" on page 21.
   - **Windows**: In the SVN Launcher, specify the path in the Display field.
4. All rendering servers in the cluster should be configured for auto-logon.
5. All values are expressed in pixels at the pixel pitch of each display device.
6. The display areas specified for rendering servers can overlap, and the pixel pitch of each tile can be different.
7. Blank lines in the wall configuration file, or lines that start with the # symbol, are ignored.

**Example: Wall configuration file to define a symmetric display**

This example of a wall configuration file defines a virtual display area of 3120 x 2048 pixels. The display area has four tiles, with each tile having an area of 1560 x 1024 pixels. SVN arranges the tiles in a 2 x 2 array with the top two tiles being separate displays on the same rendering server `render56` (see Figure 2).

```
render54:0.0  3120 2048 0      0      1560   1024
render55:0.0  3120 2048 1560   0      1560   1024
render56:0.0  3120 2048 0      1024   1560   1024
render56:0.1  3120 2048 1560   1024   1560   1024
```

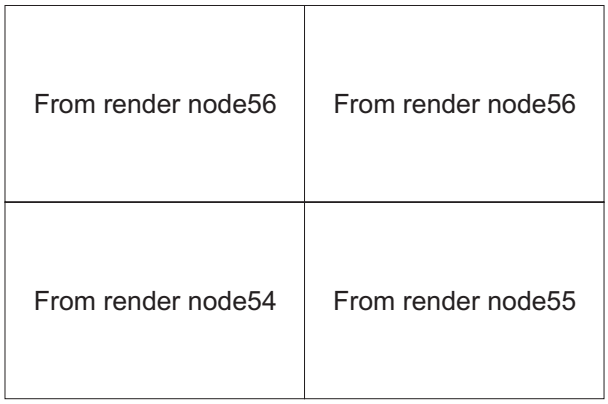



*Figure 2. Virtual display with 2 x 2 tile array*

### Sample wall configuration file

The installation procedure creates a sample wall configuration file — *wall.cfg* — in the following directories:

- **Linux**: */opt/IBM/dcv/svn/examples/linux_sample_wall.cfg*
- **Windows**: *<Program Files>\IBM\IBM Deep Computing Visualization\svn\wall.cfg*

This sample file can be used to verify the installation. You can modify the file, or create a new one, to describe the display wall configuration corresponding to your cluster configuration.

The sample file specifies a single rendering server on the local device and is loaded by default when you run the SVN Launcher. On Linux, you can specify this file as an argument to the `svn_enable` command, with the result that a single rendering server is enabled on the local device.

## SVN transport options

The network transport protocols that SVN supports are TCP/IP Sockets, Sockets Direct Programming (SDP), and OpenMPI. (Other MPI implementations — such as Scali MPI Connect, Cisco/TopSpin, and MPICH — might work with SVN but are not supported.)

## Linux considerations for SVN

You might need to configure your Linux system for SVN operations.

### X Server

Ensure that the X Servers are configured consistently across all of the rendering servers.

### Remote Shell

If you intend to use Remote Shell (RSH) for remote access to the rendering servers, the system administrator must enable it. The rendering servers and the application host must be included in the authorization files for RSH *(/etc/hosts or $HOME/.rhosts)* on each system (including the application host).

### Secure Shell

If you intend to use Secure Shell (SSH), the public key of the user on the application host must be generated and made accessible to the same user on each rendering server and to the application host. Doing this provides remote access without requiring the user to supply passwords. The key is usually stored in the `~/.ssh/authorized_keys` directory.

### GDM security

To ensure connectivity when GNOME Display Manager (GDM) security is in place, you must configure the X Server to allow TCP connections. Edit the */etc/X11/gdm/gdm.conf* file and set **DisallowTCP=false**.

# Chapter 5. Configuring RVN

Before using RVN, you need to configure the application host and end stations.

## Application hosts and end stations

RVN uses an application host to run OpenGL graphics applications and displays the output on one or more end stations that are reachable over the network.

The application host sends updates (in the form of pixel data) to each connected end station. End stations send user events (such as mouse or keyboard actions) to the host. Each end station is responsible for:

- Displaying one or more application windows
- Sending user events to the application host for processing

## RVN modes of operation

RVN can run in two different modes — VNC mode or X11 Export mode. The mode you choose depends on your operating system and hardware configuration.

- In VNC mode, the graphics output is displayed in an application window within VNC Viewer. You must use RealVNC VE to access RVN in VNC mode.
- In X11 Export mode, the graphics output is displayed in the application window on the exported display. X11 Export mode is available only on Linux systems.

A variety of configurations is possible:

- RVN in VNC mode (see Figure 3 on page 16)
- RVN in VNC mode with desktop isolation — Linux only (see Figure 4 on page 17)
- RVN in X11 Export mode — Linux only (see Figure 5 on page 18)

## RVN in VNC mode

RVN in VNC mode can be used in a single connection or in a collaborative configuration with multiple end stations.

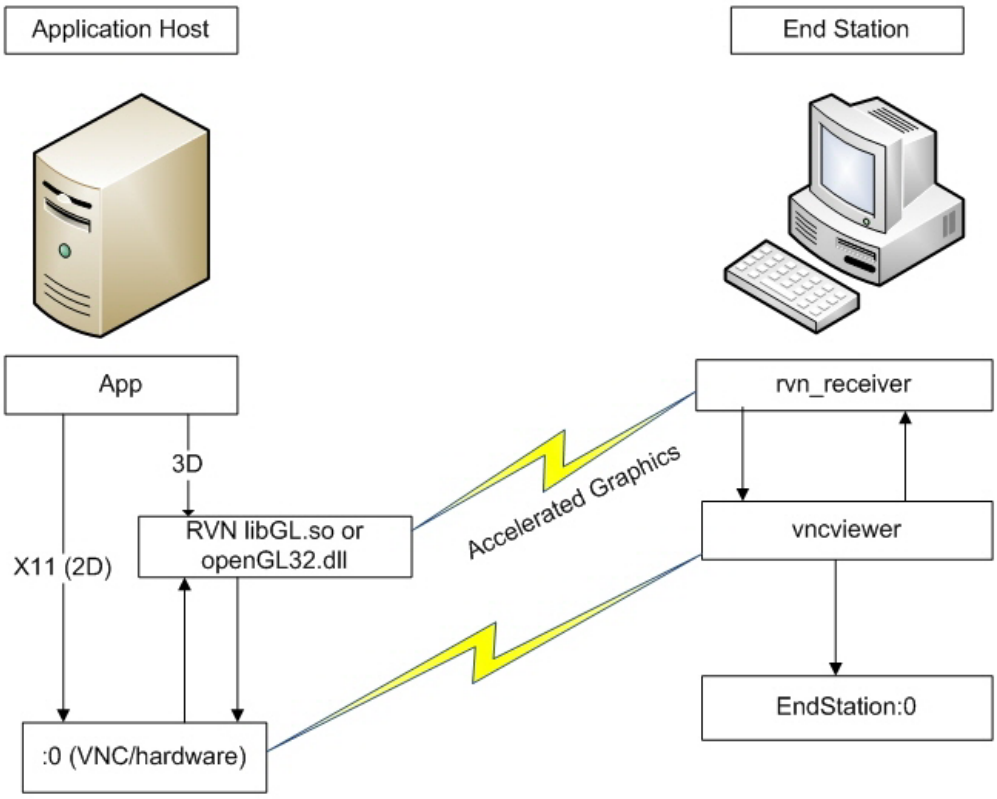



*Figure 3. RVN in VNC mode*

Figure 3 illustrates RVN in VNC mode for remote desktop functionality. The application host sends compressed pixels to the RVN end stations. It can also write 3-D data to the local application windows (when the Show on Server check box is selected).

## RVN in VNC mode with desktop isolation

RVN in VNC mode with desktop isolation provides a virtual desktop on the application host.

**Notes:**

1. The application host must be running Linux. The end stations can run Linux or Windows.
2. VNC Server uses the virtual display ID specified on the command line. If you do not specify a display, VNC Server defaults to the next available display number.

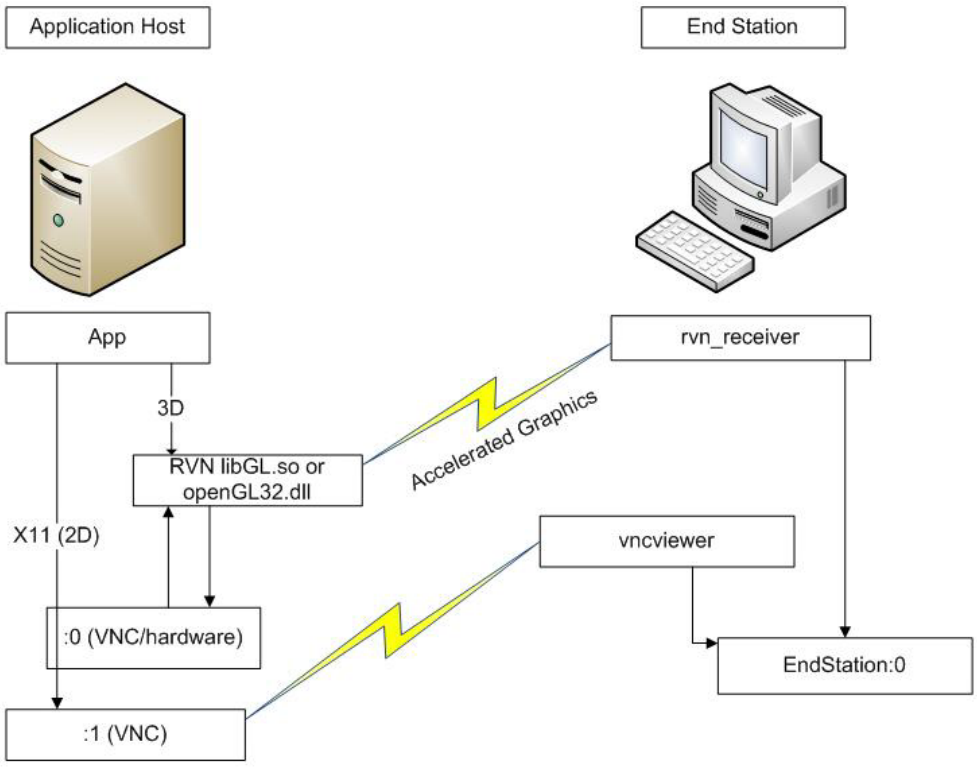



*Figure 4. RVN in VNC mode with desktop isolation*

Figure 4 illustrates RVN in VNC desktop isolation mode. In this mode, you can create multiple virtual X11 desktops on the application host for remote viewing. From an end station, you can run an application on a remote desktop using hardware-accelerated 3-D rendering.

## RVN in X11 Export mode

Use RVN in X11 Export mode when you want to export the content of a single application instead of the entire desktop.

Only one end station can connect to an application host when you use RVN in X11 Export mode. The application host sets conference codes that must be shared with the end station (you can create your own conference codes).

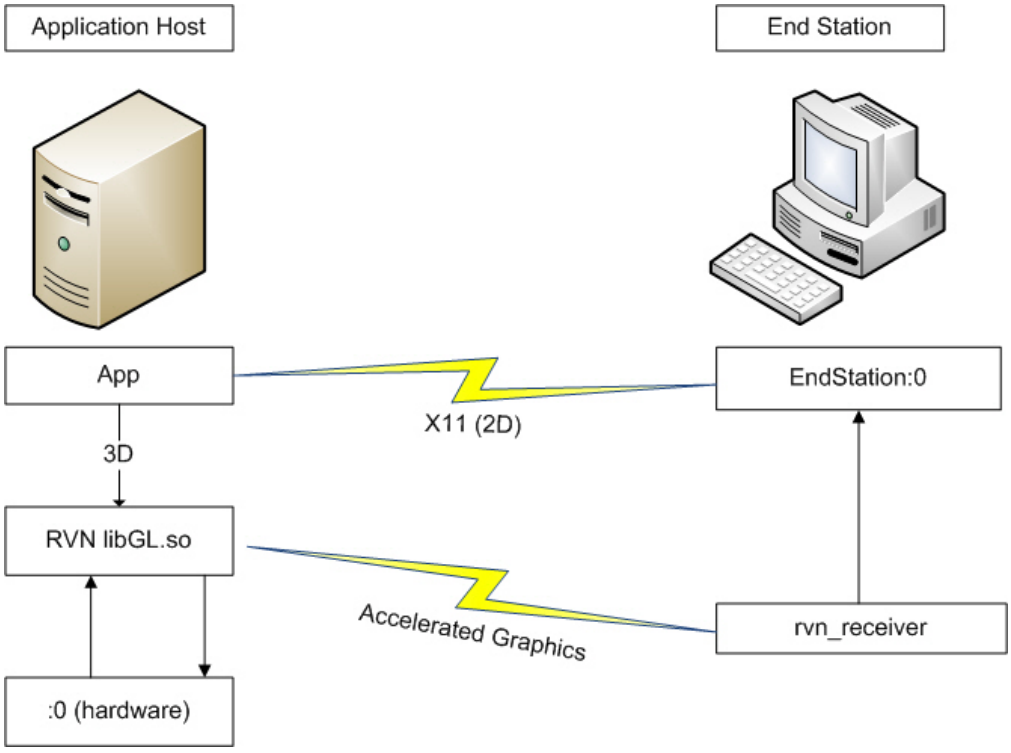



*Figure 5. RVN in X11 Export mode*

Figure 5 shows RVN in X11 Export mode. 2-D user data is transported in X11 Export mode while 3-D data is transported through RVN's accelerated graphics functionality.

**Notes:**

1. The application host and end station must be running Linux.
2. Use the RVN Launcher (and not `rvn_enable`) to start your application — this ensures correct authentication.
3. A maximum of one end station can be used in this configuration.
4. An X Server must be running on the end station.
5. X11 Export mode performs best when the application host and end station are on the same LAN.
6. Deep Computing Visualization must be installed on both the application host and the end station.

### Conference ID and key

Only one end station can connect to an application host when you use RVN in X11 Export mode. You can create your own **Conference Coordinator ID** and **Conference Coordinator Key** that must be shared by both the application host and the end station.

## The RVN coordinator

When you use X11 Export mode, a process called `rvn_coordinator` automatically manages the connection between the application host and the end stations.

The process uses TCP ports 7200-7220 — you can change these by setting the global environment variables RVN_COORDINATOR_PORT and RVN_TOP_PORT, defining the start and end ports, respectively.

**Notes:**

1. `rvn_coordinator` must run on the application host.
2. To manually start the RVN coordinator, run the `rvn_coordinator` command.

## The RVN mini-coordinator

The RVN mini-coordinator automatically manages the connection between the application host and end stations in VNC mode on Linux and Windows.

The process uses TCP ports 7300-7399 — you can change these by setting the global environment variables RVN_MINI_START_PORT and RVN_MINI_MAX_PORT, defining the start and end ports, respectively.

### UDP ports

The RVN mini-coordinator also uses the RVN_UDP_START_PORT environment variable to determine the UDP port to bind to (when Long Distance Transport is enabled). The default port is 7400 — the RVN mini-coordinator searches sequentially until it finds an open UDP port.

**Notes:**

1. The RVN mini-coordinator process is built into the RVN OpenGL library — it is never invoked manually.
2. If the RVN mini-coordinator is disabled (because the RVN_MINI_START_PORT environment variable is set to a negative value), RVN will try to use the RVN coordinator instead.

## X Server configuration

Ensure that the X Server on the application host is configured for 24-bit color depth.

The following example illustrates a pragma from an *xorg.conf* configuration file with the default 24-bit color depth:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Videocard0"
        Monitor    "Benq"
        DefaultDepth     24
        SubSection "Display"
                Depth     24
                Modes    "1920x1200_60" "1600x1200"
                         "1400x1050" "1280x1024" "1280x960"
                         "1152x864" "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Depth     16
                Modes    "800x600" "640x480"
        EndSubSection
EndSection
```

**Note:** Ensure that the RECORD extension is enabled in the X11 Server configuration files. To check if the extension is provided, use the following command:

- **xdpyinfo | grep RECORD**

If the extension is not configured, add the following line: **Load** "**record**" in the *Module* section of the */etc/X11/xorg.conf* file.

# Chapter 6. Using Scalable Visual Networking

This chapter describes how to run applications with Scalable Visual Networking (SVN).

## Using SVN on Linux

SVN is enabled or disabled on the application host with the following user-level commands:

**svn_enable**
Ensures that applications that are subsequently started on the X Server can use SVN.

**svn_disable**
Prevents applications from using SVN

**svn_query**
Displays a message indicating whether SVN is enabled.

**Note:** If Deep Computing Visualization is not enabled, run the `dcv_enable` command. (This is required only if you have disabled the software using, for example, the `dcv_disable` command.)

### The svn_enable command

The `svn_enable` command enables the use of SVN for applications that are subsequently started on the application host.

The syntax of the `svn_enable` command is as follows:

```
svn_enable [<svn_enable_options>]
```

**Note:** The *-wall* option is always required.

#### Command-line options for svn_enable

The following command-line options can be specified for the `svn_enable` command:

**-clientrender {0 | 1 }**
Enables or disables rendering on the application host. The possible values are 0 (disabled) and 1 (enabled — the default setting).

**-display** {*display name*}
Contains the display name.

If this option is not set, the default display (for example, the display specified by the DISPLAY environment variable) is used.

Example: **svn_enable -wall /tmp/wall.cfg -display dcvzpro5:0.0.** causes the display at *dcvzpro5* to be used.

**Note:** The -wall option is always required. (See the *-wall* option below.)

**-localaddr {address}**
Sets the name of the InfiniBand network interface to use for socket transport. (The default is the local host name.)

**-mpibin** {*path*}
Contains the path to MPI user commands.

**Notes:**

1. If the command that is used to start MPI is not in the PATH environment variable, use the *-mpibin* command-line option.
2. If you have more than one version of MPI installed, the directory of the version you want to use with SVN should be listed first in the LD_LIBRARY_PATH environment variable.

**-mpicomm** {*type*}
Sets the MPI transport launch script. (The default is *mpicomm.openmpi.ssh*.)

**-selector {0 | 1}**
Enables or disables the SVN Window Selector (for more information, see "The SVN Window Selector" on page 26). The possible values are 0 (disabled) and 1 (enabled — the default setting).

**-swap {0 | 1}**
Synchronizes SVN with framelocking hardware (if present).

Example: **svn_enable -wall /tmp/wall.cfg -swap 1** causes SVN to use framelocking on the display rendering servers.

**-sync {0 | 1 | 2 | 3}**
Specifies the type of application-to-display synchronization to be used:

**0** No synchronization: Rendering servers will run as fast as possible.

**1** Synchronized: Rendering servers receive new data once per frame.

**2** Synchronized: Each rendering server acknowledges receipt.

**3** Synchronized: All rendering servers are updated together.

Example: The **svn_enable -wall /tmp/wall.cfg -sync 3** command causes SVN to synchronize updates from the graphics application.

**-transport { MPI | SOCKET }**
Specifies the type of transport to use — sockets or MPI. If SDP sockets are enabled at the operating system level, they will be used instead of TCP sockets.

For example: **svn_enable -wall /tmp/wall.cfg -transport SOCKET**

**-wall** {*wall.cfg*}
Contains the path to the wall configuration file. This parameter is required. (For more information, see "Display wall configuration" on page 11.)

Example, if the wall configuration file is in your home directory and you are user dcv, use this command: **svn_enable -wall /home/dcv/wall.cfg**

## The svn_disable command

The `svn_disable` command prevents the use of SVN by applications that are started after the command is issued.

### Running an application with SVN on Linux

To run an application, perform the following steps:

1. Define how the scene geometry will be displayed by the rendering servers. Store this information in the *wall.cfg* file. (For further information, see "Display wall configuration" on page 11.)

   **Note:** This step might already have been completed by the system administrator.
2. Run the **svn_enable** command. Specify any command-line options that might be needed to define the environment.
3. Run a graphics application as normal: Double-click its icon or issue a shell command.

## Using SVN on Windows

You can start OpenGL graphics applications with SVN by using the SVN Launcher.

### The SVN Launcher

The SVN Launcher is a graphical user interface that launches applications. Figure 6 shows the Launcher window.

**Note:** The SVN Launcher is available on Windows only.

To start the SVN Launcher:

- Use the SVN Launcher shortcut: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Launcher**.

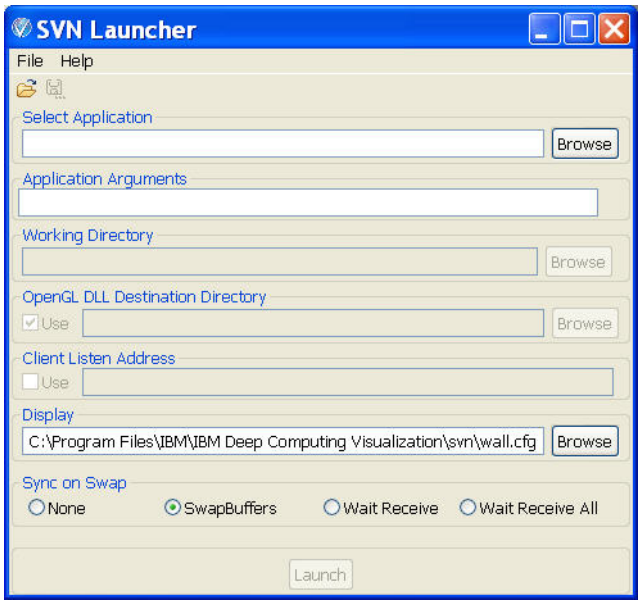



*Figure 6. The SVN Launcher*

You can specify the following options in the main window of the SVN Launcher:

**Select Application**
Enter the full path to your graphics application or click **Browse** and navigate to the application.

**Notes:**

1. You can also browse to a shortcut. The target field of the shortcut will populate the application field.
2. The application parameters and working directory in the Launcher are not updated.
3. Invalid paths are displayed in red text.

**Application Arguments**
Specify any additional arguments for the application, if required.

**Working Directory**
The directory the graphics application will run in — equivalent to the 'Starts in' parameter in a Windows shortcut. This path defaults to the application directory. To use a different directory, enter the full path or click **Browse** and navigate to the directory.

**OpenGL DLL Destination Directory**
The target directory for the SVN version of the OpenGL Dynamic Link Library (DLL) that is used by your graphics application. When the **Use** check box is selected, SVN will copy its own OpenGL DLL to this directory, providing SVN capabilities to your application. This path defaults to the application directory. To use a different directory, enter the full path or click **Browse** and navigate to the directory.

**Notes:**

1. Some graphics application use a secondary application for 3-D rendering. In this case, the home directory of the secondary application must be specified as the OpenGL DLL Destination Directory — if it is different from the location of the main application.
2. The SVN OpenGL DLL will be removed from this directory when the graphics application is terminated.
3. Some applications might work without needing to copy the SVN OpenGL DLL. In this case, you can disable the file copying procedure by clearing the **Use** check box.

**Display**
Enter the full path to the wall configuration file. This path defaults to the SVN installation directory

**Sync on Swap**
Specify the type of application-to-display synchronization to be used:

**0** None: Rendering servers will run as fast as possible.

**1** SwapBuffers: Rendering servers receive new data once per frame.

**2** Wait Receive: Each rendering server acknowledges receipt.

**2** Wait Receive All: All rendering servers are updated together.

## Saving and loading profiles

A profile is a collection of Dashboard settings for a graphics application. You can save a profile and load it later to start your graphics application with customized runtime options.

To save a profile:

- Select **File** > **Save** and name the new file, using the extension *.svn*.

To load a profile:

- Click **File** > **Open** and select the file. (You can also double-click the profile icon or shortcut to achieve the same result.) The graphics application opens with your runtime options already configured.

### The SVN Listener

The SVN Listener is a background process that detects when you launch an application using SVN and then starts the rendering servers for that application.


*Figure 7. The SVN Listener*

If the Listener is removed from the Startup folder or otherwise disabled, SVN rendering will not work. You can start the Listener manually by clicking **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Listener** or by opening the **winsvn_listen.exe** file in the *<Program Files>\IBM\IBM Deep Computing Visualization\svn* directory.

If you do not use SVN, you can safely remove the shortcut entry.

**Note:** If you change the operating system accessibility display settings (change font sizes, for example), you must restart the SVN Launcher before the changes take effect.

## Running an application with SVN on Windows

To run an application with SVN on Windows, perform the following steps:

1. Update the wall configuration file, if necessary. (A default *wall.cfg* file is stored in the IBM Deep Computing Visualization\svn folder.) Alternatively, create a custom wall configuration file and specify that file in the Launcher.

   **Note:** This step might already have been completed by the system administrator.
2. Click **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Launcher**.
3. Select or browse to an OpenGL application. Set additional configuration options, if required.
4. Click **Launch**. The application starts and the graphics output is displayed on the wall.

   **Notes:**

   a. You can also browse to a shortcut. The target field of the shortcut will populate the application field.

   b. The application parameters and working directory in the Launcher are not updated.

   c. Invalid paths are displayed in red text.

## The SVN Window Selector

You can use the SVN Window Selector to choose which 3-D window to display on the wall.


*Figure 8. The SVN Window Selector on Linux.*

The available options are:

**Select Win**
Click this button and then click the application window you want to display on the wall.

**Cycle Win**
Click this button to cycle through all open 3-D windows on the application host (Linux) or to cycle through all open 3-D windows in the associated application (Windows).

**Notes:**

1. On Linux, there is a single SVN Window Selector per desktop. On Windows, there is an SVN Window Selector per application.
2. The SVN Window Selector starts automatically.

## NVidia framelocking

Framelocking is the process of synchronizing display pixel scanning from a master system to the other systems in the network, ensuring that the displays are synchronized with each other.

**Note:** Framelocking for SVN is supported on Linux systems only.

To enable framelocking across a display cluster, perform the following steps:

1. Install a compatible graphics card with framelocking (G-Sync) capabilities on each interconnected rendering server.
2. Install appropriate drivers for the graphics cards.
3. Enable framelocking for the graphics cards.
4. Run the **svn_enable** command using the *-swap 1* command line option. This option instructs the system to use framelocking in the cluster. For more information, see “Command-line options for svn_enable” on page 21.
5. Run your graphics application. The screen refresh rates on the rendering servers will now be synchronized.

**Note:** It might also be necessary to set the *-sync 3* option to ensure that application updates, as well as physical refreshes, are synchronized. See your hardware vendor’s documentation for full instructions on how to do this.

# Chapter 7. Using Remote Visual Networking

Remote Visual Networking (RVN) has components that are common to RVN on both Linux and on Windows. This chapter describes those components and then explains how to run applications with on both operating systems.

## The RVN Launcher

You can use the RVN Launcher to launch applications.

To start the RVN Launcher:

- **Linux**: Run the **rvn_dashboard** command. For more information, see "The rvn_dashboard command" on page 33.
- **Windows**: Use the RVN Launcher shortcut from the Windows **Start** menu (**Start > All Programs > IBM > IBM Deep Computing Visualization > RVN Launcher**).

You can specify the following options in the RVN Launcher:

**Select Mode (Linux)**

**VNC mode**
RVN uses RealVNC VE to manage connections to end stations.

**X11 Export mode**
RVN uses the X Server on the end station. This mode requires the use of conference access codes.

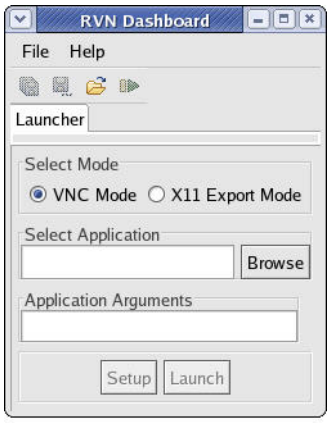



*Figure 9. The RVN Launcher on Linux with VNC mode selected*

**Select Application**
Specify the full path to your graphics application or click **Browse** and navigate to the application.

**Notes:**

1. You can also browse to a shortcut. The target field of the shortcut will populate the application field.

2. The application parameters and working directory in the Launcher are not updated.
3. Invalid paths are displayed in red text.

**Application Arguments**
Specify any additional arguments for the application.

**Working Directory (Windows)**
Defaults to the application directory — equivalent to the 'Starts in' parameter in a Windows shortcut. To use a different directory, specify the path or browse to the directory.

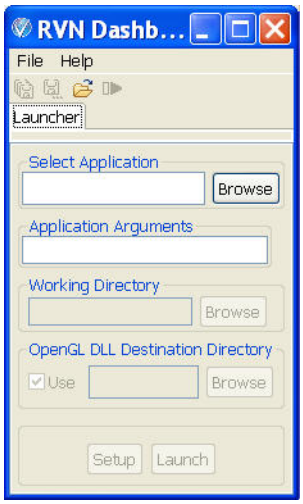



*Figure 10. The RVN Launcher on Windows*

**OpenGL DLL Destination Directory (Windows)**
The target directory for the RVN version of the OpenGL DLL that is used by your graphics application. When the Use check box is selected, RVN will copy its own OpenGL DLL to this directory, providing RVN capabilities to your application. This path defaults to the application directory. To use a different directory, enter the full path or click **Browse** and navigate to the directory.

**Notes:**

1. Some graphics application use a secondary application for 3-D rendering. In this case, the home directory of the secondary application must be specified as the OpenGL DLL Destination Directory — if it is different from the location of the main application.
2. The RVN OpenGL DLL will be removed from this directory when the graphics application is terminated.
3. Some applications might work without needing to copy the RVN OpenGL DLL. In this case, you can disable the file-copying procedure by clearing the **Use** check box


*Figure 11. The RVN Launcher on Linux with X11 Export mode selected*

**Conference ID**
Supplies the conference ID string for the RVN session.

**Conference key**
Supplies the conference access key for the RVN session. (Required for X11 Export mode.)

**Notes:**

1. The conferencing information must be shared among the participants in the conference.
2. This option overrides the `RVN_CONFERENCE_KEY` environment variable.

**Note:** For X11 Export mode, you must set the `DISPLAY` environment variable to the IP address of the end station before you start the Launcher:
**export DISPLAY=<receiver>:0.0.**
To display the Dashboard locally, set the `RVN_DASHBOARD_DISPLAY` environment variable to `:0.0`.

## Saving and loading profiles

A profile is a collection of Dashboard settings for a graphics application. You can save a profile and load it later to start your graphics application with customized runtime options.

To save a profile:

- Select **File** > **Save** and name the new file, using the extension *.rvn*.

To load a profile:

- Click **File** > **Open** and select the file. (You can also double-click the profile icon or shortcut to achieve the same result.) The graphics application opens with your runtime options already configured.

## The RVN Dashboard

You can use the RVN Dashboard to control runtime options for your graphics applications. The following figure show the main window of the RVN Dashboard.

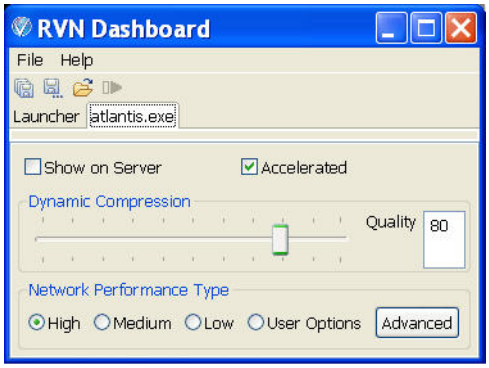



*Figure 12. RVN Dashboard main window (Windows)*

The RVN Dashboard provides the following runtime controls:

**Show on server**
Displays the 3-D graphics on the application host while also sending it to an end station. If this check box is selected, and accelerated graphics are used, RVN updates the application host screen directly. To change this setting, use the environment variable `RVN_HOST_SHOW_PIXELS` or load a saved profile with the Show on server check box selected.

**Note:** This option is available in VNC mode only.

**Accelerated**
Selected by default. Clearing this check box disables RVN and its ability to display graphics directly on the connected end stations.

**Note:** This option is available in VNC mode only.

**Dynamic Compression**
Controls the balance between image compression and image quality when the image is changing. Choosing a high value improves the quality of the dynamic images but requires more data to be sent for each image, which can reduce responsiveness or lower the frame rate. If you select 100% compression, lossless compression will be used.

**Quality**
Shows the current value for dynamic compression. Instead of using the Dynamic Compression slider, you can manually enter a value in the box.

**Network Performance Type**
Uses predefined settings for environment variables, based on network speed. To see each setting, or to create User settings, click **Advanced.**

**Note:** The network settings apply to all of the connected end stations. See Table 5 on page 31 for more information.

*Table 5. Default options for the RVN Launcher, based on network type*

| Network type | Default settings |
|---|---|
| High performance | Accelerated graphics: Selected<br>Interactive mode: Selected<br>Static image boost: Selected<br>Dynamic compression quality: 80<br>Low bandwidth: Off<br>Show Partial Frames: Off |
| Medium performance | Accelerated graphics: Selected<br>Interactive mode: Selected<br>Static image boost: Selected<br>Dynamic compression quality: 50<br>Low bandwidth: Off<br>Show Partial Frames: Off |
| Low performance | Accelerated graphics: Selected<br>Interactive mode: Selected<br>Static image boost: Selected<br>Dynamic compression quality: 20<br>Low bandwidth: Off<br>Show Partial Frames: Selected |

## Network performance options

When you click **Advanced** in the RVN Dashboard, the following network performance options are available:

- **Frame Sending Modes**

  **Interactive Mode**
  RVN compresses and sends the latest frame to the end station. Interactive Mode ensures that your graphics application output is updated at maximum speed, although some frames could be dropped by the network transport to provide the maximum possible frame rate.

  **Paced** Sends every frame that the application generates, so that users at the end stations can see all of the rendered data. Paced mode can slow the application significantly but it also reduces frame-rate jitter, where image frames are not displayed at regular intervals. This variability in the frame-to-frame interval is usually the result of network load. Paced mode introduces interval delays, causing the frames to be displayed at regular intervals.

  The slider determines the length of the delay, from 10 to 1,000 ms.

- **Quality settings**

  **Static image boost**
  Controls the balance between image compression and image quality. When this check box is selected, RVN rapidly transfers lower-quality images while the image is changing, and then updates the image with a high-quality update, thereby enhancing the quality of the static image. If you select 100% compression, lossless compression will be used.

  **Note:** Rapid flickering might occur when this check box is selected. This occurs when the application is rendering continuously but slowly and RVN has enough free bandwidth to interleave low quality and high quality images. To avoid this behavior, clear this check box.

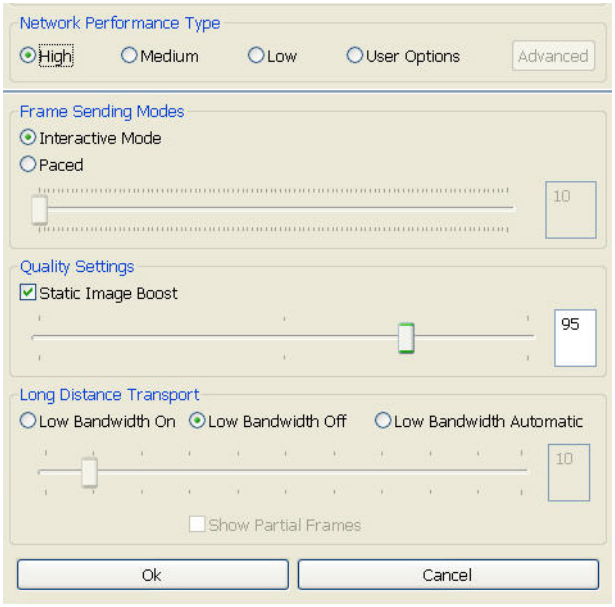



*Figure 13. The RVN Dashboard: Network performance options*

- **Long Distance Transport**

  An alternate mode for sending pixels when latency is high. Select one of the following options:

  **Low bandwidth On**
  Low bandwidth will be used.

  **Low bandwidth Off**
  Low bandwidth will not be used.

  **Low bandwidth Automatic**
  Automatically enables low bandwidth image transfer, based on throughput, latency, and packet-loss threshold.

  If you select Low bandwidth Automatic or Low bandwidth On, the following options are available:

  **Show Partial Frames**
  Select this check box to see the display in the event of significant network losses. Some data segments might be lost and older data will be used to fill any gaps.

  **Image quality slider**
  Select this check box to adjust the image quality setting when the Low Bandwidth check box is selected.

Changes to these settings are applied immediately. (When you click **OK**, the settings become the new defaults for the current user for the current application instance.) Click **Cancel** to undo any changes. You can save these user settings for later use.

**Note:** You cannot change the settings for the High, Medium, or Low network types.

**Note:** If you change the operating system accessibility display settings (for example, change font sizes), you must restart the Dashboard before the changes take effect.

## Using RVN on Linux

RVN is enabled or disabled on Linux with the following user-level commands:

**rvn_enable**
Ensures that applications subsequently started on the X Server can use RVN.

**rvn_disable**
Prevents subsequently started applications on the X Server from using RVN.

**rvn_dashboard**
Starts the RVN Dashboard.

**rvn_viewer**
Starts a VNC viewer on the end station.

**rvn_receiver**
Receives data in X11 Export mode.

**rvn_query**
Displays a message indicating if RVN is enabled.

**Note:** If Deep Computing Visualization is not enabled, run the `dcv_enable` command. (This is required only if you have disabled the software using, for example, the `dcv_disable` command.)

### The rvn_enable command

The `rvn_enable` command starts RVN on the specified display.

The command uses the current DISPLAY environment variable or gets the display name from the command line option "–display *dpyname*". The command is formatted as follows:

**`rvn_enable [-display {`*dpyname*`}]`**

### The rvn_disable command

The `rvn_disable` command removes RVN functionality. OpenGL applications that are started after this command is run do not use RVN functionality. This command uses the current DISPLAY environment variable or gets the display name as a command line option "–display *dpyname*". The command is formatted as follows:

**`rvn_disable [-display {`*dpyname*`}]`**

### The rvn_dashboard command

The `rvn_dashboard` command starts the RVN Dashboard. You can use the command to start your graphics application, change runtime settings, or configure the session to use X11 Export mode. When the Dashboard is started, any running RVN-enabled applications will connect to the Dashboard (after a brief delay).

This is the syntax of the `rvn_dashboard` command:

**`rvn_dashboard [-conf <confID>] [-key <confkey>] [appname] [app-args]`**

**Notes:**

1. The `rvn_dashboard` command replaces `rvn_sender`. (Scripts using `rvn_sender` in previous releases of Deep Computing Visualization will continue to function in this release.)
2. For a description of related command-line options, see "Command-line options for rvn_dashboard."
3. For a description of the related environment variables, see "Environment variables for rvn_dashboard" on page 54.

### Command-line options for rvn_dashboard

This section defines the options that can be specified on the `rvn_dashboard` command.

**-conf** *<confID>*
Supplies the conference ID string for an X11 Export session.

**Note:** This option overrides the `RVN_CONFERENCE_ID` environment variable.

**-key** *<confkey>*
Supplies the conference access key for an X11 Export session.

**Notes:**

1. This option overrides the `RVN_CONFERENCE_KEY` environment variable.
2. The conference information must shared with each participant before the start of the conference.
3. This information is needed only for X11 Export mode. When VNC is used, authentication is automatically generated.

## The rvn_receiver command

The `rvn_receiver` command is used in X11 Export mode to receive the pixel data.

**Notes:**

1. `rvn_receiver` communicates with the coordinator process using the default port 7200. If this port is being used by another application or if the application host is using a different port, you can override the default value by setting the `RVN_COORDINATOR_PORT` environment variable to an available port number.

This is the syntax of the `rvn_receiver` command:

```
rvn_receiver [host] <confID> <confkey>
```

The `rvn_receiver` command uses the options that were specified on the command, as well as the settings of related environment variables.

- For a description of command-line options, see "Command-line options for rvn_receiver."
- For a description of related environment variables, see "Environment variables for rvn_receiver" on page 58.

### Command-line options for rvn_receiver

The options that can be specified on the `rvn_receiver` command are:

*<host>* The name of the application host.

*<confid>*
A string identifying the RVN conference session that you are connecting to. This information must be distributed to all conference participants before

the conference starts. The person responsible for providing this information is the conference initiator (the person invoking the coordinator process).

*<confkey>*
The security password required to access the RVN conference session. This information must be sent to all conference participants before the conference starts.

## The rvn_viewer command

The `rvn_viewer` command starts the VNC Viewer on a Linux end station.

This is the syntax of the `rvn_viewer` command:

```
rvn_viewer [<servername[:portnum]>] [<other viewer options>] [-h]
```

- For a description of the command-line options, see "Command-line options for rvn_viewer."

### Command-line options for rvn_viewer

The options that can be specified on the `rvn_viewer` command are:

*-servername [ : portnum]*
Specifies the name of the server to connect to. If desktop isolation is used, this option specifies the port for the virtual session.

**-h** Displays help information for the command.

*other viewer options*
Specifies any necessary additional options.

**Note:** For more information, visit RealVNC.

## Running an application with RVN on Linux

This section describes the different ways to run applications with RVN:

- RVN in VNC mode
- RVN in VNC mode — desktop isolation
- RVN in X11 Export mode

### Running an application with RVN in VNC mode

To run an application with RVN in VNC mode, perform the following steps:

**On the application host**

1. Start the RVN Dashboard: Run the **rvn_dashboard** command.

   **Note:** You can bypass the Dashboard by running the `rvn_enable` command and then launching your graphics application.
2. Select **VNC mode.**
3. Specify the path to your graphics application. (Specify application arguments, if required.)
4. Click **Launch**. The application starts and a new Dashboard tab appears (if the RVN Dashboard is running).

**On each end station**

1. Start **VNC Viewer**.
2. Connect to the application host.

## Running an application with RVN in VNC desktop isolation mode

To run an application with RVN in VNC desktop isolation mode, perform the following steps:

**On the application host**

1. Log on to the application host and run the following command:

   `vncserver :<N> -depth 24 -pixelformat rgb888 -geometry <x> <y>`

   (where N is a unique display number, and x and y are the required desktop dimensions)

**On the first end station**

1. Start **VNC Viewer**.
2. Connect to the application host, specifying display number `<N>`. A new instance of VNC Server is started on the application host.
3. Start the RVN Dashboard: Run the **rvn_dashboard** command. (VNC mode is automatically selected.)

   **Note:** You can bypass the Dashboard by running the `rvn_enable` command and then launching the graphics application.
4. Specify the path to the graphics application. (Specify application arguments, if required.)
5. Click **Launch**. The application starts and a new Dashboard tab appears (if the RVN Dashboard is running).

**On subsequent end stations**

1. Start **VNC Viewer**.
2. Connect to the application host, specifying display number `<N>`. A new instance of VNC Server is started on the application host.

## Running an application with RVN in X11 Export mode

To run an application with RVN in X11 Export mode, perform the following steps:

**On the exported display <end station>**

1. Run the **xhost +** command.

   **Note:** You can use alternative methods for allowing X connections.
2. Run the **rvn_receiver <host> <confID> <confkey>** command.

**On the application host**

1. Run the **export DISPLAY=<end station>:0.0** command.
2. Enable RVN: Run the **rvn_enable** command.
3. Start the RVN Dashboard: Run the **rvn_dashboard** command.
4. Select **X11 Export Mode**.
5. Specify the path to your graphics application. (Specify application arguments, if required.)
6. Enter the **Conference Coordinator ID** and **Conference Coordinator Key.**
7. Click **Launch**. The application starts and a new Dashboard tab appears.

**Notes:**

1. The application host and end station must be running Linux.
2. Use the RVN Launcher (and not `rvn_enable`) to start your application — this ensures correct authentication.

# Using RVN on Windows

RVN on Windows always uses VNC to transport data between the application host and end stations.

## Running an application with RVN on Windows

To run an application with RVN on Windows, perform the following steps:

**On the application host**

1. Start the RVN Launcher: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > RVN Launcher**.
2. Specify the path to the graphics application. (Specify application arguments, if required.)
3. Click **Launch**. The application starts and the graphics output is sent to the remote display.

**On each end station**

1. Start **VNC Viewer**.
2. Connect to the application host.

# Chapter 8. SVN and RVN interoperability

Both the SVN and RVN functions of Deep Computing Visualization can operate simultaneously in a joint SVN-RVN session.

**Note:** SVN and RVN interoperability is available on Linux only.

SVN-RVN can run in the following modes:

- RVN rendering on the application host
- RVN application display on a dedicated rendering server

## RVN rendering on the application host

To use this mode you must install and configure RealVNC VE on the SVN application host. Then use the following procedure:

1. Enable SVN: Run the **svn_enable** command.
2. Enable RVN: Run the **rvn_enable** command.
3. Start your graphics application.
4. Connect to the application host from an RVN end station.
5. Start the RVN Dashboard to control the remote session parameters.

### Example

On the application host *viz01*:

```
svn_enable -wall /tmp/wall.cfg -transport SOCKET
rvn_enable
/usr/X11R6/lib/xscreensavers/queens
```

On an RVN end station:

```
rvn_viewer viz01
```

## Display on a dedicated rendering server

To use this mode, you need a dedicated rendering server. RealVNC VE must be installed and configured on the rendering server. Then perform the following steps:

1. On the SVN application host, export DISPLAY to the rendering server.
2. Enable SVN: Run the **svn_enable** command on the exported display.
3. Enable RVN: Run the **rvn_enable** command on the exported display.
4. Start your graphics application.
5. Connect to the rendering server from a standard end station.
6. Start the RVN Dashboard to control the remote session parameters.

### Example

On the rendering server *viz02*:

**xhost +**

On the application host *viz01*:

```
export DISPLAY=viz02:0.0
svn_enable -wall /tmp/wall.cfg -transport SOCKET
rvn_enable
/usr/X11R6/lib/xscreensavers/queens
```

On an RVN end station:

**rvn_viewer viz02**

# Chapter 9. Advanced users

This chapter contains specialized information for administrators and programmers.

## Programming considerations

To obtain maximum performance, you should consider the following information when you are programming applications that will run under SVN or RVN (or SVN-RVN)

**Front-buffer programs**
If a program renders to the front buffer, graphics commands might be delayed reaching the SVN rendering servers. This delay occurs while the SVN client fills its output buffers with rendering commands. SVN requires that an explicit or implicit flush occur to guarantee that the SVN rendering servers are updated on a timely basis. OpenGL commands that explicitly or implicitly cause a flush are:

- **glFlush()**
- **glXSwapBuffers() [On Windows: SwapBuffers]**
- **glFinish()**
- **glXMakeCurrent() [On Windows: wglMakeCurrent]**

**Offscreen buffer and pixmap rendering programs**
Programs that render to an off-screen buffer and then retrieve the rendered data might not work correctly. The same is true of programs that render to pixmaps.

**Non-OpenGL programs**
Programs that do not use OpenGL to produce their graphic images will not work correctly in SVN or RVN.

## OpenGL overloads with SVN

This chapter contains specialized information for administrators and programmers.

SVN uses a mechanism called *overloads* to provide enhanced functions (for example, providing stereo output for non-stereo-enabled applications). You can also create your own overloads to modify the behavior of any OpenGL function. Using these custom overloads, you can affect the behavior of specific applications during an SVN session. In addition, you can also group individual overloads together and create an *overload set*. With an overload set, the individual overloads work in sequence to achieve specific application functions.

**Notes:**

1. To use an overload or an overload set, you must load it with SVN at run time.
2. For sample overload files, look in the */opt/IBM/dcv/svn/examples/overload* directory.
3. Overloads are supported on Linux only.

All SVN overloads and overload sets work by intercepting OpenGL calls made by the graphics application. When a call is intercepted, the overload replaces the

information in the call with the modified instructions contained in the overload. The overload process works because the SVN graphics pipeline follows a specific sequence of events:

1. The SVN OpenGL library installed on the application host intercepts OpenGL calls.
2. SVN encodes the OpenGL calls and sends the encoded calls over the network to the rendering servers.
3. The rendering servers receive and decode the data.
4. Each of the rendering servers processes these commands and displays its portion of the final image.

At various stages of this pipeline you can modify the parameters of an intercepted OpenGL call. This process is called *overloading* an OpenGL call.

## The basic overload

The following example introduces a simple overload that intercepts an OpenGL call and modifies its parameters to invert colors. The mechanism for this overload functions by intercepting all calls that the target application makes to `glColor**` and modifying the associated OpenGL parameters.

**Note:** In this example, the overload version of `glColor**` refers to `glColor3f`. However, this example easily extends to similar calls.

The function template:

```
static void
o_glColor3f(float original_r, float original_g, float original_b)
{
float modified_r = 1 - original_r;
float modified_g = 1 - original_g;
float modified_b = 1 - original_b;

//call system OpenGL glColor3f passing it as arguments
//modified_r, modified_g, modified_b
}
```

SVN must fill in the missing call to the system OpenGL `glColor3f` and make sure that any application call to `glColor3f` is diverted to the overloaded version, `o_glColor3f`. Before proceeding, more detail is needed regarding SVN behavior when loading and using a custom set of overloads. The main requirement for the custom overload set is that it must implement the `Overload` function, the prototype for which is:

```
void Overload(glOp *glOperationsTable, glOp *glSystemOperationsTable);
```

When loading the custom overload set, SVN automatically calls the `Overload` function contained within the set and passes it pointers to an array of current SVN OpenGL operations and an array of the original system OpenGL operations. These arrays, especially the SVN OpenGL operations table, play a central role in the overload mechanism:

- The entry in the SVN operations table that corresponds to `glColor3f` provides a handle that calls the current SVN `glColor3f` from the overloaded version `o_glColor3f`.
- By replacing the entry in the SVN operations table that corresponds to `glColor3f` with a new entry that points to the overload version `o_glColor3f`, SVN will call the overloaded version instead of the original.

- The array of original system OpenGL operations is provided in case the newly written overload needs to make OpenGL calls using the system OpenGL library rather than the current (possibly overloaded) versions that SVN is using.

To access these arrays, SVN provides a series of numeric constants, and each of these constants corresponds to an OpenGL call. For example, `glOperationsTable[GLCOLOR3F]` provides access to the entry that corresponds to `glColor3f`. All these constants have the form of the OpenGL call they are referring to, and the reference *must* be written in capital letters. In addition, you must include the file `wire.h` in the source file for the overload module.

**Note:** The `wire.h` file is located in the `$SVN_ROOT/include/wire.h` directory.

Using these requirements, you can complete the overload code file:

```
#include "wire.h"
static void(*s_glColor3f)(float,float,float);
static void   o_glColor3f(float,float,float);

//this function will be automatically called by SVN:
void Overload(glOp *glOperationsTable, glOp *glSystemOperationsTable)
{
  // save a handle to the system glColor3f(...) call:
  s_glColor3f = (void(*)(float,float,float)) glOperationsTable [GLCOLOR3F];

  //replace the entry with a handle to the overloaded version:
  glOperationsTable[GLCOLOR3F] = (glOp)o_glColor3f;
}

static void
o_glColor3f(float original_r, float original_g, float original_b)
{

  float modified_r = 1 - original_r;
  float modified_g = 1 - original_g;
  float modified_b = 1 - original_b;

  (*s_glColor3f)(modified_r,modified_g,modified_b);
}
```

**Note:** For this overload to operate, the overload `o_glColor3f(..)` calls the OpenGL version of the `glColor3f(..)` function with modified parameters. Although this is the common method, such behavior is not always required. In some cases, an overload might need to call different OpenGL methods or maybe none whatsoever. As long as the overload produces the desired output, this is perfectly acceptable.

After you finish writing the overload, you need to configure SVN to load and use the code. Follow these steps:

1. Compile the overload code into a named, shared object (for example, *invert_colors.so*).

   **Note:** You can use the sample *Makefile* provided with the SVN code.

2. Create a file named *overload_files.txt* that contains a list of overload modules for SVN to use.

   **Notes:**

   a. The `overload_files.txt` file has one line for each overload module.

   b. The format for each entry has the form *<absolute_path>/invert_colors.so*.

3. Set the $SVN_OVERLOAD_FILE environment variable to point to overload_files.txt.

   **Note:** This variable should specify the full path to the text file.

After you complete these steps, run SVN on an application that uses glColor3f. You can use the provided *rotvp* test application for this purpose. All the colors in the application should be the opposite of the normal colors displayed during standard operation.

## System interface for overloads

The basic overload described in the previous section performs the same in all cases. However, more complex overloads (like the one in this section) might need to differentiate between SVN instances. In this case, the overload set must obtain information that is specific to the calling instance of SVN (the application host or one of the rendering servers). This is accomplished through an internal SVN function DVGetDisplayName() that returns the name of the X display used by the calling instance of SVN. The signature of this function is as follows:

```
char *DVGetDisplayName();
```

**Note:** Overload functions will be called by all SVN instances, client and servers.

This call returns a pointer to a string that is set as follows:

- On an SVN server, the return value of DVGetDisplayName() points to a string containing the name of the X display that it outputs to.
  - This string has the format *server_name[:d.s]*.
  - The display name is identical to the one found in the wall configuration file.
- On the SVN client, the return value of DVGetDisplayName() points to a static string containing the word "client".
- If the DVGetDisplayName() call fails, the result will be NULL.

To call this function, an overload set must use the glOperationsTable array method required for making system OpenGL calls. The entry in this array corresponding to index DVGETDISPLAYNAME provides a handle to the function:

```
char*(*DVGetDisplayNameHandle)() =
  char*(*)()) glOperationsTable[DVGETDISPLAYNAME];
char* myDisplayName = DVGetDisplayNameHandle();
```

**Note:** The DVGETDISPLAYNAME constant is also declared in the file *$SVN_ROOT/include/wire.h*. The wire.h file needs to be included in the overload source code file.

For example, to invert application colors on the SVN servers only, the Overload(...) function is:

```
void Overload(glOp *glOperationsTable, glOp *glSystemOperationsTable)
{
char*(*DVGetDisplayNameHandle)() =
  char*(*)()) glOperationsTable[DVGETDISPLAYNAME];
char* myDisplayName = DVGetDisplayNameHandle();

if (myDisplayName==NULL)
{
 fprintf(stderr,"Failed to obtain SVN system info;\n");
return;
}
else if ( !strcmp(myDisplayName,"client") )
```

```
// where "client" is the name of the application host
{
 fprintf(stderr,"SVN client: aborting color inversion overload...;\n");
return;
}

fprintf(stderr,"SVN server rendering on display %s
 using color inversion overload... \n", myDisplayName);

s_glColor3f = (void(*)(float,float,float)) glOperationsTable [GLCOLOR3F];
glOperationsTable[GLCOLOR3F] = (glOp)o_glColor3f;
}
```

Another example of differentiating between SVN instances is the Stereo Overload set. By using this set, you can display a regular (non-stereo) OpenGL application in stereo vision (active or passive). This is accomplished as follows:

- The application runs on the client rendering server and displays on two SVN rendering servers, one for left eye and one for right eye.
- Each server uses the Stereo Overload set to output either a left eye or a right eye view for stereo viewing.
- SVN delivers the output from the servers into a stereo projection system.
- The SVN client can have different behaviors:
  - It can be used instead of one of the servers displaying the left eye or right eye image.
  - It can bypass the overload set and display the standard (non-stereo) application output.

To obtain left and right eye views through the Stereo Overload set, SVN applies the following methods:

- The overload set on each SVN instance obtains its own display name by using a handle to the function `DVGetDisplayName()`. You can then specify whether the output should be left eye, right eye, or non-stereo for each display.
- To produce a left or right eye image, a stereo transformation needs to be added to the OpenGL projection matrix. Therefore, the Stereo Overload set overloads all OpenGL calls that affect the projection matrix and makes sure that the stereo transformation is always applied.

When you are using the Stereo Overload set, you follow the steps that are used for a simple overload, plus some additional steps.

- These are the basic steps that are required for all overload sets:
  1. Compile the overload code into a named, shared object (for example, *stereo_overload.so*).

     **Note:** You can use the sample *Makefile* provided with the SVN code.
  2. Create a file named *overload_files.txt* that contains a list of overload modules for SVN to use.
     - The `overload_files.txt` file has one line for each overload module.
     - The format for each entry has the form *<path>/stereo_overload.so*.
  3. Set the `$SVN_OVERLOAD_FILE` environment variable to point to `overload_files.txt`.

     **Note:** This variable requires the full path to the text file.
- In addition to the basic steps, the Stereo Overload set also requires the following steps:

1. Set up a stereo configuration file.
   - This file should contain one entry for each display that will be used for stereo vision.
   - The entry uses the form:

     *server_name[:d.s]* {*STEREO_LEFT*|*STEREO_RIGHT*}

     **Note:** You must use the exact server and display names that are used in the wall configuration file.
2. If the *client* is to be used for stereo vision, add an entry using the form:

   *client* {*STEREO_LEFT*|*STEREO_RIGHT*}
3. Set the environment variable `$SVN_STEREO_FILE` to point to the stereo configuration file.

   **Note:** This variable requires the full path to the text file.
4. Run the desired OpenGL application through SVN and feed the left eye and right eye displays to stereo projection hardware.

**Notes:**

1. If an entry corresponding to any of the SVN displays is missing from the stereo configuration file, the SVN instance running that display simply bypasses the Stereo Overload set and produces standard (non-stereo) output.
2. The steps listed in this section are specific to this particular implementation of the Stereo Overload set. They are not typically required by SVN.
3. Other developers might implement the same functionality through overloads using other methods to determine whether to output left eye, right eye, or a standard image on a particular display.

## Using multiple overload sets

The previous example described an overload *list file* with one entry that pointed to the associated overload shared object. The overload list file can also have more than one entry. In such a case, SVN loads and uses all the provided overload sets. For example, you could use both sets developed in the previous examples. If you did so, the resulting display would have stereo images with inverted colors. To accomplish this, you would create an overload list file that would look like this:

```
<path>/stereo_overload.so
<path>/invert_colors.so
```

**Notes:**

1. You must update the `$SVN_OVERLOAD_FILE` environment variable to point to the overload list file.
2. You could create a single overload with multiple functions. However, if you use an overload list file that points to multiple overloads, you can define specific end results by combining different overloads in an overload list file.
3. Although you can achieve positive results with multiple overloads, you can also create system conflicts if you are not careful with how you combine overloads in the overload file list. Review the overload list file and the associated overloads to confirm that you will achieve the preferred results.

The following example illustrates what happens when two overload sets try to overload the same OpenGL call. In this case, overload_set_1 and overload_set_2 both overload `glColor3f(..)`. This is done using the following code fragments:

**overload_set_1**

```
void Overload(glOp *glOperationsTable, glOp *glSystemOperationsTable)
{
  s_glColor3f = (void(*)(float,float,float)) glOperationsTable [GLCOLOR3F];
  glOperationsTable[GLCOLOR3F] = (glOp)o_1_glColor3f;
}
static void
o_1_glColor3f(float r, float g, float b)
{
   ...
   (*s_glColor3f)(...)
   ...
}
```

**overload_set_2**

```
void Overload(glOp *glOperationsTable, glOp *glSystemOperationsTable)
{
  s_glColor3f = (void(*)(float,float,float)) glOperationsTable [GLCOLOR3F];
  glOperationsTable[GLCOLOR3F] = (glOp)o_2_glColor3f;
}
static void
o_2_glColor3f(float r, float g, float b)
{
   ...
   (*s_glColor3f)(...)
   ...
}
```

**Note:** For this example, the overload list file contains two entries:

```
<path>/overload_set_1.so
<path>/overload_set_2.so
```

When SVN loads the overload sets, these actions occur:

1. SVN calls the `Overload(...)` function of `overload_set_1` and passes it a pointer to the `glOperationsTable`.
2. `overload_set_1` stores the handle to `glColor3f(...)` that it finds in `glOperationsTable`.
3. `overload_set_1` replaces the handle to `glColor3f(...)` with a handle to its own `o_1_glColor3f(...)`.
4. SVN calls the `Overload(...)` function of `overload_set_2` and passes it a pointer to the `glOperationsTable`.
5. `overload_set_2` stores the handle it finds in `glOperationsTable`. However, that handle was replaced by `overload_set_1` and it is now a handle to `o_1_glColor3f(...)`.

   **Note:** As a result, whenever `overload_set_2` is using the handle it found in `glOperationsTable`, `overload_set_2` is actually calling `o_1_glColor3f(...)`.
6. `overload_set_2` replaces the handle from `glOperationsTable` with a handle to its own `o_2_glColor3f(...)`.
7. Because the handle in `glOperationsTable` was replaced twice, SVN calls are diverted to `o_2_glColor3f`.

Figure 14 on page 48 shows the effect that multiple overload sets have when they make application calls to the same `glColor3f(...)` handle. The illustrated pipeline might be broken if, for example, `o_2_glColor3f(...)` would not attempt to use the handle it receives from SVN and stores in `s_glColor3f(...)`. In general, when multiple sets overload the same OpenGL call, that call triggers the execution of the overload function from the last set of overloads contained in the overload list.

Then, depending on the implementation of each overload, the call might propagate to the overload version of the same function defined in a set that is higher in the list.

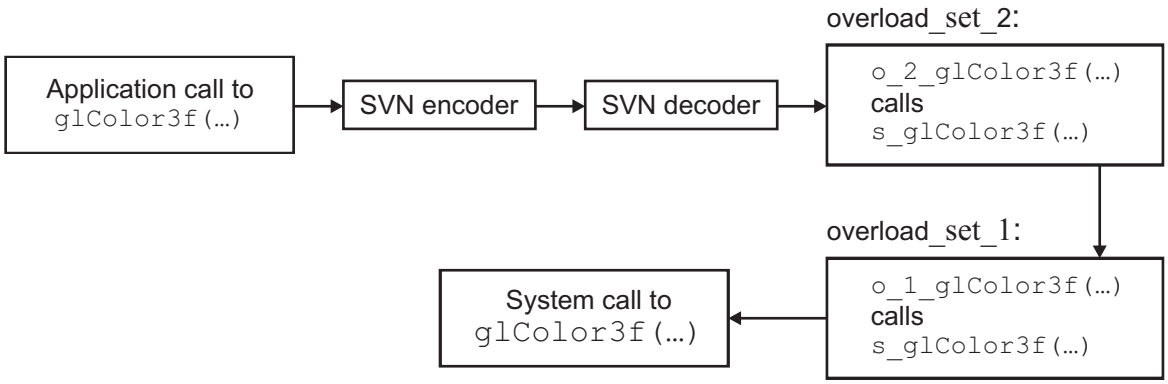



*Figure 14. Multiple overload sets*

# Chapter 10. Diagnostics

This chapter describes problems that might occur with SVN and RVN. See the following sections for diagnostic information:



**Notes:**

1. Appendix B, "Error Messages" provides additional diagnostic information.
2. If you cannot determine the cause of failure, request the assistance of the IBM Field Support Center.
3. Before contacting IBM for service, see the README files provided on the Deep Computing Visualization page listed under "Related information" on page xi.

## SVN problems

If you encounter any problems using SVN on Linux, set `SVN_VERBOSE=`*1* to preserve any log files that are generated. (Use SOCKET transport for best performance.) The log files are stored in the system temporary directory (*/tmp*) on the application host and the rendering servers.

If the `svn_enable` command exits with an error, examine the error message and follow the suggested action. If that does not solve the problem, use local problem-reporting procedures. Consider the following possibilities:

- If the problem seems to be on the server side, check that the required libraries and executable files are installed on the rendering server.
  - If needed, reinstall the libraries and executable files, and follow local problem-reporting procedures if the system requires additional diagnostics.
- If **RSH** is used to start the remote tasks, remember that it does not run the user profile *$HOME/.bash_profile*. However, **RSH** does run the *$HOME/.bashrc* file. This can affect which library paths are actually searched.
  - If needed, update the library paths defined in the *$HOME/.bashrc* file, and follow local problem-reporting procedures if the system requires additional diagnostics.

If the application runs but nothing is displayed on the server after the initial banner:

1. Try enabling the `-selector` option on the `svn_enable` command and selecting the display window several times.
2. Try running an OpenGL application.
3. Reboot and try the operation again.
4. If the problem persists and the system requires additional diagnostics, follow local problem-reporting procedures.

If performance is slow, use a tool such as *glxinfo* to check that accelerated rendering is being used. Check also that a high-performance network is being used by both the application host and the rendering servers.

If the server or application halts or generates a core dump:

1. Verify that the application runs correctly as a stand-alone application on the application host.
2. Collect the failure information and the core dump.
3. Contact IBM Service.

If one of the rendering servers reports that it cannot open the display, make sure that the server has enabled all X connections by running the `xhost +` command. If you receive an application message indicating that it cannot open the display:

- Verify that you have enabled the application to access the display through the `xhost +` command (or equivalent authorization commands). See also "GDM security" on page 13.
- If you cannot get the application to open the display, follow local problem-reporting procedures.

If you receive a `permission denied` message:

- Review the appropriate `RSH` or `SSH` documentation and ensure that the access for `RSH` or `OpenSSH` has been set up.
- Make sure that the pass-phrases have been supplied to the SSH agents on the application host and on all of the rendering servers, as described in the `SSH` documentation.
- If the SVN Launcher fails to start an application on Windows, the SVN Listener might have been disabled on one or more of the rendering servers. To run the Listener: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Listener**.
- If the application starts on Windows but there is no display on the rendering servers, ensure that the OpenGL Destination Directory is set correctly for the application.
- If you are still receiving the `permission denied` message, follow local problem-reporting procedures.

**Attention:** If you cannot determine the cause for any of the problems listed above, contact IBM Service.

## RVN problems

If the application host reports that it cannot export the display, make sure that the X Server has enabled all X connections by running the `xhost +` command. If you receive an application message indicating that it cannot open the display:

1. Verify that you have enabled the application to access the display through the `xhost +` command (or equivalent authorization commands). You might need to configure the X Server to allow TCP connections. Edit the */etc/X11/gdm/gdm.conf* file and set "DisallowTCP=false". See also "GDM security" on page 13.
2. If you cannot get the application to open the display, follow local problem-reporting procedures.

If you encounter any problems using RVN, isolate the component causing the problem by varying the configuration as described in the following list. If any of these actions causes the system to issue a message, follow the actions suggested by the message. If a command is not working correctly, verify that the proper command-line options and environment variables are set (see Chapter 7, "Using Remote Visual Networking," on page 27). If these actions do not solve the problem, follow local problem-reporting procedures.

1. Use only a single end station.
2. Run without RealVNC VE (Linux only).
3. Run with RealVNC VE, but without enabling the accelerated graphics function.
4. If the application starts on Windows but no new Dashboard tab appears on the Launcher, ensure that the OpenGL Destination Directory is set correctly for the application.
5. If the Windows end station does not display the 3-D graphics, ensure that the system PATH environment variable on the end station contains the correct RVN binary directories.
6. Run the installation verification procedure. See "Verifying Installation" on page 9.

## X Server display problems

If you are having problems related to X Server display, check the following information:

- `/var/log/XFree86.0.log` or `/var/log/Xorg.0.log`
  - This file contains information about X.
- **xdpyinfo**
  - This display utility provides information for X.
- **xhost + vis80**
  - Show X windows on the display at `vis80`.
- Run an OpenGL application
  - If you are using the correct 3-D accelerated drivers, you should get more than 10,000 frames per second.

## Display problems

If you are having display problems, check the following issues:

- Confirm that the `DISPLAY` environment variable is correctly set.
- Confirm that you can invoke `xclock` and that it appears on the intended display.
  - If this fails, the display owner might need to issue the `xhost + localhost` command.
- From the server, verify that you can you access the local `DISPLAY (:0)`.
  - If this fails, the display owner might need to issue the `xhost + localhost` command.
  - The display owner might also need to issue the `chmod a+rw /dev/nv*` command.

### Flickering

Your display might flicker because the application is rendering continuously but not quickly enough. To resolve this, clear the Static Image Boost check box.

## NVIDIA display problems

If you are having NVIDIA display problems, check the following directories and take the suggested corrective actions if needed:

- cat /proc/driver/nvidia/version

  - Verify that the correct driver level is installed, and reinstall it if necessary. For information about the correct driver level, see "SVN prerequisites" on page 3. If you reinstall display drivers, you will need to run the `dcv_enable` command.
- cat /proc/driver/nvidia/cards/0
  - Verify that the reported card is a supported display adapter. If the adapter is not supported, install a supported card and its associated driver.
- cat /proc/driver/nvidia/agp/card
  - Verify that you are receiving the expected output. If not, check the NVIDIA documentation for steps to resolve the problem.
- cat /proc/driver/nvidia/agp/host-bridge
  - Verify that you are receiving the expected output. If not, check the NVIDIA documentation for steps to resolve the problem.
- cat /proc/driver/nvidia/agp/status
  - Verify that you are receiving the expected output. If not, check the NVIDIA documentation for steps to resolve the problem.
- If these actions do not solve the problem, follow local problem-reporting procedures.

# Appendix A. Environment variables

You can set environment variables to customize SVN and RVN commands. To set an environment variable:

- **Linux**
  1. Open a terminal window.
  2. Enter **export** *{environment variable} = <value>*.

  **Note:** These environment variables are valid for OpenGL applications that you start in the terminal.
- **Windows**
  1. Open the System Properties control panel applet: Click **Start > Control Panel > System**.
  2. Click the **Advanced** tab.
  3. Click the **Environment Variables** button.
  4. Enter a new environment variable or update an existing variable to the required value.

  **Note:** These settings are permanent (until they are manually removed).

For more information, see these sections:



## Environment variables for SVN on Linux

In addition to SVN command-line options, you can also set the following environment variables to control SVN processing. These variables must be defined in the environment on the application host.

**Notes:**

1. The following definitions assume that you are using the Bash shell.
2. The environment variables listed in this section should be exported.
3. Where applicable, the underlined value is the default.

**SVN_BANNER_COLOR=***X_display_color_name*
This environment variable specifies the color used for the initial server-banner display. The default is `SkyBlue`.

**SVN_BANNER_FONT=***X_display_font_name*
This environment variable specifies the font used by the rendering X Server to display the banner text. The default is `variable`.

**SVN_BANNER_TIME=***display_time_in_seconds*
When the rendering server is started, it displays a banner showing the overall geometry and its portion of it. This environment variable controls how long the banner will be displayed. The default is 5 seconds.

**SVN_CLIENT_OVERLOAD_FILE=***overload-file-path*
This environment variable identifies SVN overloads that will be loaded in the client application process. It overrides the SVN_OVERLOAD_FILE setting for the specified process. For more information about overloads, see "OpenGL overloads with SVN" on page 41.

**SVN_FIRST_WINDOW_ONLY={0 | 1}**
This environment variable provides a solution for an NVIDIA graphics driver framelocking limitation. When this variable is set to 1, it instructs SVN to send information to the rendering servers that relates only to the first window that was opened by the application, thus allowing it to be correctly framelocked .

**SVN_INITIAL_SCALED_WINDOW=***n*
When you set this variable, it instructs SVN to initially display the nth OpenGL window that was opened by the application, rather than the first window that was opened (which is the default behavior). Use the SVN Window Selector to specify which window is shown on the display wall.

**Note:** If you set this environment variable, the SVN_FIRST_WINDOW_ONLY environment variable is ignored.

**SVN_OVERLOAD_FILE=***overload-file-path*
This environment variable identifies a set of overloads that will be loaded into all SVN processes, on both the application host and the rendering server. For more information about overloads, see "OpenGL overloads with SVN" on page 41.

**SVN_SERVER_OVERLOAD_FILE=***overload-file-path*
This environment variable identifies SVN overloads that will be loaded in the SVN server processes. It overrides the SVN_OVERLOAD_FILE setting for the specified process. For more information about overloads, see "OpenGL overloads with SVN" on page 41.

**SVN_SWAP_ON_RETRACE ={ 0 | 1 }**
This environment variable delays synchronization until the displays retrace on the server.

**Note:** If you are using NVIDIA graphics G-Series adapters, setting SVN_SWAP_ON_RETRACE to 1 enables the framelock function.

**SVN_VERBOSE={ 0 | 1 }**
This environment variable performs the following functions:

- Turns on echoing of each line in SVN for debugging purposes
- Saves intermediate work files for analysis and troubleshooting.

## Environment variables for rvn_dashboard

In addition to the options listed in "The RVN Launcher" on page 27 and in "Command-line options for rvn_dashboard" on page 34, you can also define the following environment variables to control **rvn_dashboard** processing.

**RVN_ALTERNATE_VISUALS=1**
Set this environment variable only if the OpenGL application fails when this variable is not set.

**Note:** This environment variable is not normally used, and it might be required only when you are using an X Server with limited function. It asks for a 24-bit true-color non-stereo visual, irrespective of what the application tried to request.

**RVN_CONFERENCE_ID=***confID*
Specifies the name of the RVN conference session. (X11 Export mode only)

**Notes:**

1. This information is needed only for X11 Export mode. When you are using VNC, this information is automatically generated during VNC authentication.
2. This information must be sent to each conference participant before the start of the conference.

**RVN_CONFERENCE_KEY=***confkey*
Specifies the key required to access the specified RVN conference session. (X11 Export mode only)

**Notes:**

1. This information is needed only for X11 Export mode. When you are using VNC, this information is automatically generated during VNC authentication.
2. This information must be sent to each conference participant before the start of the conference.

**RVN_COORDINATOR_PORT=***port*
Specifies the TCP port on the application host used by `rvn_dashboard`, `rvn_receiver`, and `rvn_coordinator` to establish an RVN conference session. This environment variable must have the same value for all three processes (`rvn_dashboard`, `rvn_receiver`, and `rvn_coordinator`) participating in the RVN conference.

**Notes:**

1. The default port is 7200.
2. This information must be sent to each conference participant before the start of the conference.

**RVN_EXTERNAL_TRANSPORT=[1|0]**
When this environment variable is set to 1, the RVN pixel backchannel is enabled and OpenGL output is transferred to the viewer more quickly.

**Note:** The default value for this environment variable is 1.

**RVN_HOST_SHOW_PIXELS=[1|0]**
Enables or disables displaying pixels on the local display of the application host in addition to sending it to the RVN end station.

**Note:** The default value for this environment variable is 1.

**RVN_IMAGE_QUALITY=[1...100]**
Controls the balance between image compression and image quality when the image is changing. The range is a percentage from 1 to 100. Choosing a high quality value (a large number) improves the visual quality of the dynamic images, but requires more data to be sent for each image, which can reduce responsiveness or lower the frame rate. If you select 100% compression, lossless compression will be used.

**Note:** The default value for this environment variable is 80 (100 is the highest quality).

**RVN_INTERACT_MODE=[1|0]**
Specifies the mode for sending frames from the application to the end station display.

- If you specify 1 (Interactive Mode), the application will run as fast as possible, always transmitting the most recently generated frame to the end station display. In doing so, it might drop stale images before trying to compress and send them, in favor of sending newer images. Select this mode if you are most interested in always viewing the most recent frame, and not concerned with seeing every generated frame.
- If you specify 0 (Paced Mode), RVN sends every frame that the application generates and, in so doing, can often significantly slow the application. Users at the end stations can see all of the rendered data. Paced Mode creates a regular, paced viewing experience.

**Note:** The default value for this environment variable is 1.

**RVN_LOCAL_DISPLAY=***display*
The default hardware-rendering adapter is :0. Use this environment variable to specify non-default hardware.

**RVN_PACING_TIME=[10...1000]**
Specifies the length of delay to be used for pacing mode, which is selected by setting `RVN_INTERACT_MODE=0`. The time is specified in milliseconds. Increase the value until the frame rate becomes sufficiently smooth and pleasing to the eye.

**Note:** The default value for this environment variable is 10.

**RVN_QUALITY_UPDATE=[1|0]**
Sends a higher quality image when the image is not being rapidly updated by the application. When this option is selected, RVN rapidly transfers lower-quality images when the image is changing. When rendering is idle, RVN updates the display with a high-quality update, thereby enhancing the quality of the static image.

**Notes:**

1. The default value for this environment variable is 1.
2. In some circumstances, rapid flickering occurs when `RVN_QUALITY_UPDATE` is enabled. To avoid this condition, do not enable this variable.

**RVN_SUBSAMPLING=[1|2|4]**
Set this environment variable to adjust image quality through pixel subsampling.

- Specify 1 for 411 subsampling, which uses every other x and every other y pixel. 411 subsampling creates a compressed image by reducing the number of bits to approximately half, and so is likely to perceptibly decrease image quality.
- Specify 2 for 422 subsampling, which uses every other pixel in the horizontal direction. A choice of 422 subsampling generally provides sufficient quality for human perception.
- Specify 4 for 444 subsampling, which uses all pixels. A choice of 444 subsampling approximately doubles the data requirements.

**Note:** The default value for this variable is 2.

**RVN_SYSTEM_OPENGL_LIB=***complete_path_to_system_openGL_library*
Specifies the path to the `System_OpenGL` library.

**RVN_UDP=[0|1|2]**
Determines how pixels are sent when latency is high. This environment variable can be used to provide a more responsive view when interactive mode is being used (`RVN_INTERACT_MODE` is set to 1) and a sequence of dynamic frames is being sent over a low bandwidth. Specify one of the following options:

**0** Indicates that the low bandwidth option will not be used.

**1** Indicates that the low bandwidth option will be used.

**2** RVN automatically determines whether to use the low bandwidth option, based on throughput, latency, and packet-loss threshold.

**Note:** The default value for this environment variable is 0.

**RVN_UDP_QUALITY=[1...100]**
Adjusts the image quality when `RVN_UDP` is set to `ON` or `AUTO`. Because fewer frames are transmitted when a low bandwidth is used, you might want to reduce the quality of the image to increase the compression and, therefore, speed. Low values signify high compression and lower image quality; high values mean little compression and high image quality.

**Note:** The default value for this environment variable is 10.

**RVN_UDP_SHOW_PARTIAL=[1|0]**
Indicates whether to display partially updated frames. Set this variable to 1 to see a "frameless" display in the event of significant network losses, and to use old data to fill occasional gaps in the image when some data segments are lost.

**Note:** The default value for this environment variable is 0.

**RVN_UNIQIFY_CONFERENCE_ID=[1|0]**
(X11 Export mode only) Set this environment variable to 1 if you are running an OpenGL application that forks multiple processes that each load the OpenGL library, libGL.so. This ensures that each application process that is forked after the initial application process is assigned a unique conference ID, based on the conference ID for the session followed by a sequential number. For example, if the specified conference ID is `abc123`, the initial application process will use that conference ID, the second application process will use the conference ID `abc123_1`, the third will use `abc123_2`, and so on.

**Notes:**

1. This variable is relevant only for sessions that use X11 Export. It is not needed for sessions that use VNC.
2. The default value for this variable is 0.

**RVN_UPDATE_QUALITY=[1...100]**
Controls the balance between image compression and image quality when the image is not changing. The range is a percentage from the current dynamic compression quality to 100. If you select 100% compression, lossless compression will be used.

**Note:** The default value for this environment variable is 80 (100 is the highest quality).

**RVN_USE_VNC=[1|0]**
When set to 1, tells RVN to look for a VNC window when writing application images.

**Note:** On a Linux application host, the default value for this environment variable is 1. (RVN uses VNC mode.)

## Environment variables for rvn_receiver

These are the environment variables that you can set to control **rvn_receiver** processing.

**RVN_CONFERENCE_ID=***confID*
Specifies the name of the conference session.

**Note:** If `RVN_CONFERENCE_ID` and `RVN_CONFERENCE_KEY` are not specified, default values are used. As a result, the connection is not secure.

**RVN_CONFERENCE_KEY=***confkey*
Specifies the key required to access the specified conference session.

**Note:** If `RVN_CONFERENCE_ID` and `RVN_CONFERENCE_KEY` are not specified, default values are used. As a result, the connection is not secure.

**RVN_COORDINATOR_PORT=***port*
Specifies the port used by `rvn_dashboard`, `rvn_receiver`, and `rvn_coordinator` to establish an RVN session. This environment variable must have the same value for all three processes (`rvn_dashboard`, `rvn_receiver`, and `rvn_coordinator`) participating in the RVN conference.

**Notes:**

1. The default port is 7200.
2. This information must be sent to each conference participant before the start of a conference.

## Environment variables for rvn_coordinator

You can set the following environment variables to control **rvn_coordinator** processing.

**RVN_COORDINATOR_PORT=***port*
Specifies the TCP port used by `rvn_dashboard`, `rvn_receiver`, and `rvn_coordinator` to establish an RVN session. This environment variable must have the same value for all three processes (`rvn_dashboard`, `rvn_receiver`, and `rvn_coordinator`) participating in the RVN conference.

**Notes:**

1. The default port is 7200.
2. This information must be sent to each conference participant before the start of a conference.

**RVN_TOP_PORT=***top-port*
Specifies the highest TCP port number used by `rvn_coordinator` when allocating ports for use in an RVN conference. This value must be higher than the value of `RVN_COORDINATOR_PORT`.

**Note:** The default value is 7220.

## Environment variables for the RVN mini-coordinator

You can set the following environment variables to control **RVN mini-coordinator** processing.

**RVN_MINI_START_PORT=***port*
Specifies the TCP port used by the RVN mini-coordinator to establish an RVN session. A negative value disables the mini-coordinator and RVN will attempt to use the external coordinator process.

**Note:** The default port is 7300.

**RVN_MINI_MAX_PORT=***top-port*
Specifies the highest TCP port used by the RVN mini-coordinator. This value must be higher than the value of `RVN_MINI-START_PORT`.

**Note:** The default value is 7399.

**RVN_UDP_START_PORT**
Specifies the UDP port used by the RVN mini-coordinator for UDP transport. For more information, see "The RVN mini-coordinator" on page 19.

**Note:** The default value is 7400.

# Appendix B. Error Messages

When an error occurs, messages are displayed to help diagnose and solve the problem. These messages include:



## Installation messages

**Deep Computing Visualization License only requested. Product install is not attempted.**

**Explanation:** You specified the environment variable `DCV_LICENSE_ONLY` as 1 so only the license acceptance process is run. This is appropriate if you are installing the executable files on a shared file system and the user only needs to license additional rendering servers for execution.

**User response:** If you intended to install Deep Computing Visualization, reset the `DCV_LICENSE_ONLY` environment variable and run the install command again.

**Deep Computing Visualization License agreement not accepted. Product install is abandoned.**

**Explanation:** You have not responded yes to the prompt asking if the license is to be accepted.

**User response:** If you intended to install Deep Computing Visualization, you must rerun the install command and accept the license agreement when prompted.

## RVN error messages

**RVN: Memory allocation fails**

**Explanation:** RVN was unable to allocate needed memory.

**User response:** Examine system memory usage. If unable to diagnose the problem, contact IBM Service.

**RVN: Error: Compression failed to initialize**

**Explanation:** Internal error.

**User response:** Contact IBM Service.

**Deep Computing Visualization/RVN License agreement file not found on node** *host name*

**Explanation:** A copy of the license agreement file could not be found on the application host.

**User response:** Ensure that RVN is correctly installed.

**RVN: unable to open display:** *display_name*

**Explanation:** RVN was unable to open the specified display.

**User response:** Check the value of the `DISPLAY` environment variable and ensure that access has been granted through a mechanism such as `xhost` or `Xauthority`.

**RVN: Compression encoder returns error**

**Explanation:** An error was returned by the RVN internal compression encoding mechanism.

**User response:** Contact IBM Service.

---

**RVN: error <*n*> writing to receiver**

**Explanation:** An error was detected when trying to write to the end station.

**User response:** Ensure that no network issues exist, then contact IBM Service.

---

**RVN: unable to create shared memory segment**

**Explanation:** An error was returned when attempting to create a shared memory segment.

**User response:** Ensure that the system has sufficient memory and that old shared memory instances have been correctly cleaned up. If there are no memory issues, contact IBM Service.

---

**RVN: unable to attach shared memory segment**

**Explanation:** An error was returned when attempting to attach a shared memory segment.

**User response:** Contact IBM Service.

---

**RVN: unable to XShmAttach**

**Explanation:** An error was returned when attempting to call XShmAttach.

**User response:** Check if your system has enough shared memory segments and that shared memory is configured in the X server. Otherwise, contact IBM Service.

---

**RVN: receiver error reading handshake**

**Explanation:** An error was detected when trying to read from the sender.

**User response:** Ensure that no network issues exist, then contact IBM Service.

---

**RVN: compression decoder returns error**

**Explanation:** The RVN internal compression decoder returned an error.

**User response:** Contact IBM Service.

---

**RVN: read call returns error** *err*

**Explanation:** Attempting to read from socket returns error *err*.

**User response:** Contact IBM Service.

---

**RVN: Bad magic number** *number*

**Explanation:** RVN received a corrupt packet.

**User response:** Contact IBM Service.

---

**RVN: sender (***client_id***:***rvn_component_id***): error writing handshake**

**Explanation:** Internal error.

**User response:** Contact IBM Service.

---

**RVN: sender (*client_id*:*rvn_component_id*): error reading handshake fd *file_descriptor***

**Explanation:** Internal error.

**User response:** Contact IBM Service.

---

**RVN: read <num> calls returns error <n>**

**Explanation:** Internal error.

**User response:** Contact IBM Service.

---

**RVN: receiver read error**

**Explanation:** Internal error.

**User response:** Contact IBM Service.

---

**RVN: receiver setup failure**

**Explanation:** Internal error.

**User response:** Contact IBM Service.

---

**RVN: sender is rejected by coordinator due to duplicate conference ID**

**Explanation:** The conference ID is not valid.

**User response:** Use a different conference ID.

---

**RVN: Failed to initialize resource control - you should configure RECORD X11 extension**

**Explanation:** The RECORD extension is not configured for X Server.

**User response:** Add an appropriate entry for the RECORD extension to the X configuration file.

---

**RVN: Cannot init local display.**

**Explanation:** RVN was unable to open the specified display.

**User response:** Ensure that the environment value or command-line option is correct and that access to the display has been granted using the `xhost` command.

---

**RVN: Cannot init second connection to display**

**Explanation:** RVN was unable to open the specified display.

**User response:** Ensure that the environment value or command-line option is correct and that access to the display has been granted using the `xhost` command.

---

**RVN: Connected downlevel Deep Computing Visualization end station**

**Explanation:** You are using an older version of Deep Computing Visualization on the RVN end station.

**User response:** Upgrade to the latest version of Deep Computing Visualization for increased compatibility and performance.

---

**RVN: UDP send problem**

**Explanation:** This is an internal error.

**User response:** Contact IBM Service.

---

**RVN: Request to track window failed**

**Explanation:** RVN could not track the window.

**User response:** Contact IBM Service.

## SVN runtime messages

**SVN: Failed to allocate memory**

**Explanation:** SVN was unable to allocate needed memory.

**User response:** Examine system memory usage. If unable to diagnose the problem, contact IBM Service.

**SVN: SVN_DISPLAY not set. Cannot find wall configuration file, Exiting**

**Explanation:** The wall configuration file is missing or is not readable (by the current user).

**User response:** Create a wall configuration file and specify it using the DISPLAY environment variable. You can also specify it as a parameter on the `svn_enable` command (Linux) or in the SVN Launcher (Windows).

**SVN: SVN-RVN not supported on Windows**

**Explanation:** SVN-RVN Interoperability mode works on Linux only.

**User response:** Run SVN-RVN Interoperability mode on Linux.

**SVN: cannot create transport**

**Explanation:** There is a problem with the network. If unable to diagnose the problem, contact IBM Service.

**User response:** Check your network settings.

**SVN: cannot get my host name**

**Explanation:** There is a problem with the network.

**User response:** Check your network settings, DNS lookup, and the Host file.

**SVN: unable to open display**

**Explanation:** DMX is not running.

**User response:** Ensure that DMX is running.

## SVN application host messages

**Attention SVN: **** UNKNOWN FORMAT IN GL*xxxxxxx* ******
**Attention SVN: **** UNKNOWN TYPE IN GL*xxxxxxxx* ******
**Attention SVN: Unknown type in gl*xxxxxxxx*....ignoring call**

**Explanation:** The current version of Deep Computing Visualization is intended to support a specific level of OpenGL, plus some extensions. However, some of the more advanced options might not be fully supported. The application has probably used an option in an OpenGL call that is not supported by SVN.

**User response:** Contact IBM Service.

---

**Error SVN: XQueryTree failed**
**Error SVN: XGetGeometry failed**
**Error SVN: unable to open display**
**Error SVN: unable to create window**
**Error SVN: select error in selector**

**Explanation:** The SVN client has encountered an internal error trying to create the selector window (`-windowselector = 1`).

**User response:** Verify that the `DISPLAY` environment variable is set correctly. If `DISPLAY` is correct and the message still occurs, you might be able to work around the problem by disabling the selector window. If that does not help, contact IBM Service.

---

**Attention SVN: typeSize using default statement: type = <value> We are returning 4 bytes**

**Explanation:** An OpenGL datatype was not recognized. The application has probably used an option in an OpenGL call that is not supported by SVN.

**User response:** If your application does not encounter any problems, the 4 byte option is probably a good guess. However, if you are having trouble with your application after encountering this message, contact IBM Service.

# SVN rendering server messages

**Attention SVN: unresolved OpenGL op called:** <*OpenGL_call*>

**Explanation:** The current version of Deep Computing Visualization is intended to support a specific level of OpenGL, plus some extensions. (For information about the supported level of OpenGL, see “SVN prerequisites” on page 3.) However, some of the more advanced calls might not be fully supported. The application has probably used an OpenGL call that is not supported by SVN.

**User response:** Contact IBM Service.

---

**SVN: glBitmap does not support GL_UNPACK_ALIGNMENT != 1**

**Explanation:** SVN does not support this feature.

**User response:** Modify the application, if possible, to avoid use of this feature.

---

**Attention SVN: X Color** <*color*> **not found. Using white**

**Explanation:** This color is not available on the SVN rendering server.

**User response:** Modify the rgb.txt file in your X11 installation to include the missing color.

---

**SVN: No conforming visual exists on server**
**==========**
**Visual Attributes Requested**
**==========**

**Explanation:** This is a header that indicates some visual was not available in the server. The specific visual attributes will be listed.

**User response:** Ensure that the X Server on the rendering server is configured with the missing capabilities.

---

**SVN: Texture ID conflict.**

**Explanation:** A texture with the same ID has been defined previously.

**User response:** Modify the application, if possible, to use unique texture IDs.

---

**Attention SVN: Unable to allocate cell for color** <*color*>

**Explanation:** The window selector colors might be affected.

**User response:** The selector window will continue to work.

---

**SVN: Texture ID** *texture_id* **could not be found for deletion.**

**Explanation:** The indicated texture_ID could not be found.

**User response:** Ensure that the application is not attempting to delete an invalid texture.

---

**Error SVN: Host addrmode** <*address mode*> **does not match server addrmode** <*address mode*> **on server** <*server*>

**Explanation:** The application host and rendering server are not running compatible versions of the operating system. One is 32-bit and one is 64-bit.

**User response:** Install compatible versions of the operating system and try the operation again.

---

**Error SVN: Host version** <*version*> **does not match server version** <*version*> **on server** <*server*>

**Explanation:** The application host and rendering server are not running compatible versions of Deep Computing Visualization.

**User response:** Install compatible versions of Deep Computing Visualization on the rendering servers and try the operation again.

---

**Error SVN: unable to create decode thread**

**Explanation:** A server was unable to create one of the required service threads.

**User response:** Contact IBM Service.

---

**Error SVN: unable to open wall configuration file** <*file*>

**Explanation:** The file specified by the message cannot be opened by the server.

**User response:** Check that the file exists and is readable.

---

**Error SVN: not enough entries in wall configuration file** <*file*>

**Explanation:** SVN counts the number of active entries in the file, and tells each server which entry to process. The server has not found the entry it is looking for.

**User response:** Contact IBM Service.

---

**Error SVN: invalid X display string in wall configuration file** <*file*>

**Explanation:** The server name in the wall configuration file should be in the form `server:d.s.`

**User response:** Correct the entry and try the operation again.

---

**Error SVN: unable to open display:** <*display*>

**Explanation:** The indicated display is unknown or cannot be initialized.

**User response:** Check that the `xhost +` command has been issued on the node.

---

**Error SVN: BAD OPCODE <n>**

**Explanation:** The indicated number is not in the range of the numbers associated with the supported OpenGL calls. However, because the client and server use the same set of numbers, this probably means that a prior call was incorrectly processed by the server.

**User response:** Contact IBM Service.

---

**Error SVN: something bad happened at line** <*line*> **in routine** <*routine*>

**Explanation:** Internal error at indicated location.

**User response:** Contact IBM Service.

---

**Attention SVN: NVIDIA framelock is not supported on the graphics hardware on** <*node*>

**Explanation:** SVN is attempting to use framelock between graphic adapters to synchronize screen updates but the hardware on the specified node does not support framelock.

**User response:** Set the environment variable `SVN_SWAP_ON_RETRACE` to 0.

---

**Error SVN: Error querying MaxSwapGroupsNV on** <*node*>

**Explanation:** Internal error trying to get information about the framelock capabilities of the hardware.

**User response:** Set the environment variable `SVN_SWAP_ON_RETRACE` to 0. If this is not acceptable, contact IBM Service.

---

**Error SVN: Insufficient number of swap groups** <*groups*> **and/or swap barriers** <*barriers*>

**Explanation:** Probable internal error in trying to use the framelock capabilities of the hardware.

**User response:** Set the environment variable `SVN_SWAP_ON_RETRACE` to 0. If this is not acceptable, contact IBM Service.

# Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept LJEB/P905
2455 South Rd.
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States or other countries or both:

- IBM
- IBM logo
- IntelliStation®
- System x™

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel® is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

# Glossary

## B

**backchannel.** A second data path between the application host and an end station. A backchannel uses data compression to efficiently transport pixel images.

## D

**Deep Computing Visualization.** A software solution from IBM providing enhanced graphics capabilities to standard OpenGL-based graphics applications running on Linux and Windows.

**display wall.** A large display surface made up of multiple displays. Display walls are usually assembled from individual components, each of which shows a portion of the picture. A display wall can be configured as a set of 'tiles' on which several projectors are focused on a different portion of the wall. The tiles overlap slightly to avoid the visual impact of a sharp edge between them.

**DMX.** Distributed Multihead X (DMX) provides multi-head support for multiple displays attached to different systems (each of which is running a typical X Server). Visit http://dmx.sourceforge.net/ for more information.

## E

**end station.** A remote display that receives data from an application host.

## F

**framelocking.** The ability of the several graphic adapters driving a display wall or multi-adapter monitor to synchronize their screen updates so that all portions of the display change simultaneously. This is usually accomplished by having one of the adapters send the update signal to cause all the adapters to update their screens.

## I

**InfiniBand.** A specification for high-performance adapters and switches used for communications within a cluster environment.

**intercept library.** A library that is loaded as an alternate to the library or DLL normally used by a graphics application — the OpenGL library, for example. The intercept library provides the same functions as the standard library but provides extra functions, such as routing data to alternate locations.

## M

**Message Passing Interface (MPI).** An industry-standard application programming interface (API) for message exchange between two or more cooperating tasks. Widely used in high-performance and scientific computing applications.

**MPI.** See *Message Passing Interface*.

## O

**OpenFabrics.** An open source standardized Linux–based InfiniBand software stack which is hardware- and vendor-independent.

**OpenFabrics Enterprise Distribution (OFED).** OFED is a validated version of the open source OpenFabrics software stack.

**OpenGL.** An API for representing three-dimensional objects and models. Originally developed by Silicon Graphics, Inc. (now SGI, Inc.), it has become an industry standard for high-performance, 3-D graphics programming.

**Open Message Passing Interface (OpenMPI).** A message-passing library used for parallel and distributed applications.

## P

**pass-phrase.** An extension of the concept of password, but allowing a phrase, rather than a single word, to be handled. Pass-phrase is used and recommended by Secure Shell.

**port forwarding.** See *tunneling*.

## R

**remote desktop.** A display's entire desktop is shown on a remote display so that the user can interact with that desktop.

**Remote Visual Networking (RVN).** A component of Deep Computing Visualization in which the image of a three-dimensional graphic object is efficiently transmitted to a remote display running on an end station.

**rendering server.** A computer with a display which receives a set of graphic elements and a description of the desired geometry coverage, and creates (renders) a corresponding image.

**RPM Package Manager (RPM).** A tool for installing software on Linux systems.

**RVN.** See *remote visual networking*.

# S

**Scalable Visual Networking (SVN).** A component of Deep Computing Visualization in which the elements of a three-dimensional graphic object are broadcast to a cluster of displays, such as a display wall.

**scene geometry.** The location, dimensions and orientation of all the objects being represented by a three-dimensional application.

**Secure Shell (OpenSSH).** A method of authenticating and encrypting data between two computing systems that provides a high level of security. OpenSSH is a version of the Secure Shell (SSH) protocol that is distributed with Linux systems.

**single process, multiple data (SPMD).** A form of parallel computing in which several copies of the same program are started simultaneously, but each copy works with different data, and might be executing different instructions. Used by SVN.

**SVN.** See *scalable visual networking*.

# T

**tile (n.).** One of several non-overlapping, rectangular divisions of a display screen.

**tile (v.).** To arrange multiple windows so that they appear side by side and top to bottom.

**tunneling.** A technique provided by SSH which allows secure access to a remote computer by a variety of applications. Also called *port forwarding*.

# V

**visualization.** The act or process of putting into or interpreting in visual terms or in visual form.

# W

**wall configuration file.** A computer file that describes the components of a display wall. An entry in the wall configuration file identifies the component (computer) responsible for updating a particular tile of the display.

**wall display.** See *display wall*.

# Index

# P



# R



# S

# T



# U



# V



# W



# X

# Reader's comments – We'd like to hear from you

**Deep Computing Visualization**
**Installation and User Guide**
**Version 1 Release 4**

**Publication No. G224-7622-00**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.

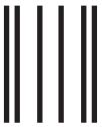
If you would like a response from IBM, please fill in the following information:

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | E-mail address |

**Readers' Comments — We'd Like to Hear from You**
G224-7622-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P181
2455 South Road
Poughkeepsie NY 12601-5400 USA

Fold and Tape | **Please do not staple** | Fold and Tape

G224-7622-00

Cut or Fold
Along Line

Printed in Ireland


G224-7622-00