Deep Computing Visualization

Installation and User Guide Version 1 Release 4.0–2

Fifth edition (April 2009)

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, DCV.

The document describes how to use the software to:

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

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 Programmable 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/systems/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 1.4.0–2

Changes in this edition include:

- **New information:**
  - Startup procedures on Windows
  - RVN Console
  - SVN Console
  - SVN Listener on Linux
  - Xinetd configuration on Linux
- **Updated information:**
  - RVN Verification on Windows
  - SVN Verification on Windows
  - SVN and RVN interoperability section has been renamed to CVN interoperability
- **Deleted information:**
  - SVN Launcher
  - SVN Dashboard
  - RVN Launcher
  - RVN Profiles
  - RVN Dashboard

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 Deep Computing Visualization. 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 support for running OpenGL applications on IBM® System p5® servers, IBM eServer™ p5 servers, and IBM eServer pSeries® servers capable of running SLES Linux Version 9 and using Gigabit Ethernet
  - 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 collaborative and scalable 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:

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

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 and outputs them to the local display as a video signal.

The Remote Visual Networking (RVN) component of Deep Computing Visualization can use low-speed wide area networks (WANs) and high speed local area networks (LANs) to provide efficient and secure transport of image data to remote displays.

The Scalable Visual Networking (SVN) component of Deep Computing Visualization uses high-speed local area networks (LANs) to link a rendering cluster to an ultra-high resolution display

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.

- When using RVN, the scene geometry and graphics state are rendered locally on a central server, and pixels are sent to one or more remote displays. This is called *Remote Visual Networking* or RVN.
- When using SVN, the scene geometry and graphics state are transmitted to a cluster of rendering servers each of which renders a portion of the overall display and native resolution. This is referred to as *Scalable Visual Networking* or SVN.

Remote Visual Networking (RVN)

Remote Visual Networking (RVN) provides remote, collaborative access to OpenGL graphics applications even across slow networks and in high(er) latency situations.

RVN intercepts the graphics commands in the rendering pipeline of an OpenGL application on a given machine, uses local resources in order to render the application, and sends the finished images to remote displays (*end stations*). It permits the use of 3-D applications even on high-latency and low-bandwidth networks. Because it uses local rendering resources, RVN provides end stations with the ability to display large complex interactive 3-D applications and data irrespective of the 3D rendering capabilities of the endstation itself. RVN can also send images to multiple destinations, providing simultaneous interaction by multiple collaborators.

Scalable Visual Networking (SVN)

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

SVN virtualizes the rendering pipeline for a graphics application over a high-speed LAN. An SVN intercept OpenGL library is installed on the application host machine running the 3-D visualization application. The library accepts the OpenGL calls made by the application, encodes them, and broadcasts the 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 at full native resolution as a tile on a display wall or projection system resulting in an extremely large multi-mega pixel display.

Collaborative Visualization Networking (CVN) Interoperability

Both the SVN and RVN functions of Deep Computing Visualization can operate simultaneously in a joint Collaborative Visualization Networking (or CVN) session.

For more information see

- Chapter 5, “Using Remote Visual Networking,” on page 21
- Chapter 7, “Using Scalable Visual Networking,” on page 35
- Chapter 8, “Collaborative Visual Networking (CVN),” on page 45

Chapter 2. Prerequisites for installation

This chapter lists the hardware and software prerequisites for installing Deep Computing Visualization.

RVN prerequisites

RVN requires the following hardware systems:

- an application host machine capable of running the requisite OpenGL application
- one or more remote machines (end stations), each of which is connected to an output display device
- a network connection (WAN/LAN) between the application host and the end station

Table 1 lists the hardware requirements for RVN.

Table 1. Hardware requirements for RVN

Device	Requirement
Application host	One of the following systems: • IBM System x 3755 • HC10
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 2 lists the software requirements for RVN.

Table 2. Software requirements for RVN

Software	Level
Operating system	Red Hat Enterprise Linux 4.0, Update 7 (32-bit or 64-bit) Microsoft Windows XP Professional Service Pack (SP) 2 (32-bit or 64-bit)
Virtual Network Computing (VNC) • VNC Server on the application host • VNC Viewer on the end stations	RealVNC Visualization Edition (RealVNC VE) **Note:** RealVNC VE is the only VNC software that is supported.
NVIDIA video driver (application host only) Modify */etc/X11/xorg.conf* to associate adapter ports with X Windows servers.	• **Linux:** 173.14.12 • **Windows:** 169.96 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.

Table 2. Software requirements for RVN (continued)

Software	Level
Xinetd server (Linux only)	Should be started

32–bit and 64–bit operating system support for RVN

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.

RealVNC EE license keys also work/enable RealVNC VE

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

SVN prerequisites

SVN requires the following hardware systems:

- An application host machine 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 **For Example**, Infiniband or Gigabit Ethernet.

Table 3 lists the hardware requirements for SVN.

Table 3. Hardware requirements for SVN

Device	Requirement
Application host	One of the following systems: • IBM System x 3755 • HC10
Rendering servers	To be decided
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 4 lists the software requirements for SVN.

Table 4. Software requirements for SVN

Software	Level
Operating system	Red Hat Enterprise Linux 4.0, Update 5 (32-bit or 64-bit) Microsoft Windows XP Professional Service Pack (SP) 2 (32-bit or 64-bit)
NVIDIA driver Modify */etc/X11/xorg.conf* to associate adapter ports with X Windows servers.	• **Linux:** 173.14.12 • **Windows:** 169.96 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.
Xinetd server	**Linux**

32-bit and 64-bit operating system support for SVN

1. For all machines the cluster must be running either 32–bit operating systems or 64–bit operating systems. Mixing of 32–bit operating systems and 64–bit operating systems is not supported.
2. SVN supports 32-bit applications running on 64-bit operating systems.
3. If you are using MPI Transport on 64-bit Linux systems, ensure that both 32-bit and 64-bit MPI libraries are installed and configured.
4. 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.
5. Xinetd server must be installed for Linux.

Optional Components

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

- **MPI (Linux only)**

  Protocol for parallel computation. The only supported implementation is OpenMPI. For more information, see http://www.open-mpi.org. Use of the Red Hat OPENMPI package is recommended.

- **DMX (Linux only)**

  Displays attached to different systems (each of which is running an X Server). For more information, see http://dmx.sourceforge.net.

Firewall configuration

Both RVN and SVN are network aware technologies and may require extra firewall configuration. Your firewall should allow traffic on the following default TCP ports:

2007 SVN Listener (default port, others may be specified)

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.

Note:

1. These ports can be changed through UI components or environment variables.
2. For SVN, it is recommended that all machines in the cluster (as well as the application host machine) 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.

Chapter 3. Installing Deep Computing Visualization

About this task

See the following sections for OS-specific installation instructions:

- "Installing on Linux application hosts (RVN and SVN) and rendering servers (SVN only)"
- "Installing on Windows application hosts and rendering servers" on page 9
- "Installing on Linux end stations" on page 8
- "Installing on Windows end stations" on page 9

Note:

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. On 64-bit Linux systems, 32-bit versions of the software must be installed before 64-bit versions.
3. On 64-bit Windows systems, both 32-bit and 64-bit versions of the software are installed as required.
4. The Windows installer also installs the Microsoft Visual C++ 2005 Redistributable package and the Intel IPP Integrated Performance Primitives Runtime package. Uninstalling either of these independent components will prevent Deep Computing Visualization from working.
5. The Deep Computing Visualization intercept libraries 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)
6. To install service packs and updates on Windows, it may be necessary to disable Deep Computing Visualization functionality temporarily. To achieve this:
   a. run the **dcv_disable** command in the bin folder under the Program Files installation folder
   b. complete the service pack setup
   c. re-enable Deep Computing Visualization using the **dcv_enable** command.
7. For the latest information about Deep Computing Visualization, including FAQs, go to http://www.ibm.com/systems/deepcomputing/visualization.

Installing on Linux application hosts (RVN and SVN) and rendering servers (SVN only)

About this task

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

1. Back up your wall configuration file and then run the **rpm -e** command to uninstall the previous packages.
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 only.

- **dcv-1.4.0-2.rh<n>.i686.rpm**
- **dcv64-1.4.0-2.rh<n>.x86_64.rpm** (if you require a 64-bit installation)

Note: **<n>** = 3,4 and corresponds to the version of RHEL.

Note: If you are using the product CD, mount the CD media to a suitable location.

3. Change the directory (**cd**) to the directory containing the RPMs.
4. Run the following command:

   ```
   rpm –i dcv.license_acceptance-1.4.i686.rpm dcv-1.4.0-2.i686.rpm
   ```

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

   ```
   rpm –i dcv64-1.4.0-2.x86_64.rpm
   ```
6. For RVN, you also need to install RealVNC VE (Select full/complete installation). See the README file for instructions on obtaining this version of RealVNC.
7. To complete the installation, see "Verifying Installation" on page 11.

Note:

1. Deep Computing Visualization is automatically enabled during the installation process. (This result is equivalent to running the `dcv_enable` command.)
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. SVN launches the rendering servers via xinetd as user 'nobody' by default. This user generally does not have access to the /dev/nvidia devices. To enable correct behavior and accelerated graphics on the rendering servers, it is advisable to edit /etc/xinetd.d/svnlisten and to change the user entry to a user with access to these devices.

Installing on Linux end stations

About this task

Note: The full installation above includes all the functionality of the end station install.

To install on a Linux system that will be used as an RVN end station, log on as a root user and 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-1.4.0-2.i686.rpm**

   Note: If you are using the product CD, mount the CD media to a suitable location.
3. Change the directory (**cd**) to the directory containing the RPM.
4. Run the following command:
   - **rpm -i dcv.endstation-1.4.0-2.i686.rpm**

5. You also need to install RealVNC VE Viewer. See the README file for instructions on obtaining this version of RealVNC.
6. To complete the installation, see "Verifying Installation" on page 11.

   Note: You do not need to run the license acceptance rpm

Installing on Windows application hosts and rendering servers

About this task

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

1. If you are migrating from an earlier release, back up your wall configuration file and then uninstall previous packages.
2. Navigate to the folder containing the installation files.
3. Run the **DCV-setup-1_4_0_2.exe** installer and follow the standard instructions for installing Windows applications.

   Note: You are prompted to install the Microsoft Visual C++ 2005 Redistributable package and the Intel IPP Integrated Performance Primitives Runtime Installer package. 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. For RVN, you also need to install RealVNC VE (Select full/complete installation). See the README file for instructions on obtaining this version of RealVNC.
5. To complete the installation, see "Verifying Installation" on page 11.

What to do next

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 consoles, respectively, so that you can load presets.

Installing on Windows end stations

About this task

Note: The full installation above includes all the functionality of the end station install.

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_0_2.exe** installer and follow the standard instructions for installing Windows applications.
4. Log off and log on again. This action updates the system path so that you can begin using the software.

5. You also need to install RealVNC VE Viewer. See the README file for instructions on obtaining this version of RealVNC.
6. To complete the installation, see "Verifying Installation" on page 11.

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

File System Minifilter driver

As part of the Deep Computing visualization product a File System Minifilter driver is installed. This driver (ibmoglrd) redirects all reads of the system *opengl32.dll* to the Deep Computing Visualization implementation of the opengl dll (*ibmdcogl.dll*). You can check if this driver is installed by running *fltmc.exe* at the command prompt. If Deep Computing Visualization is installed you should see an output similar to the following:

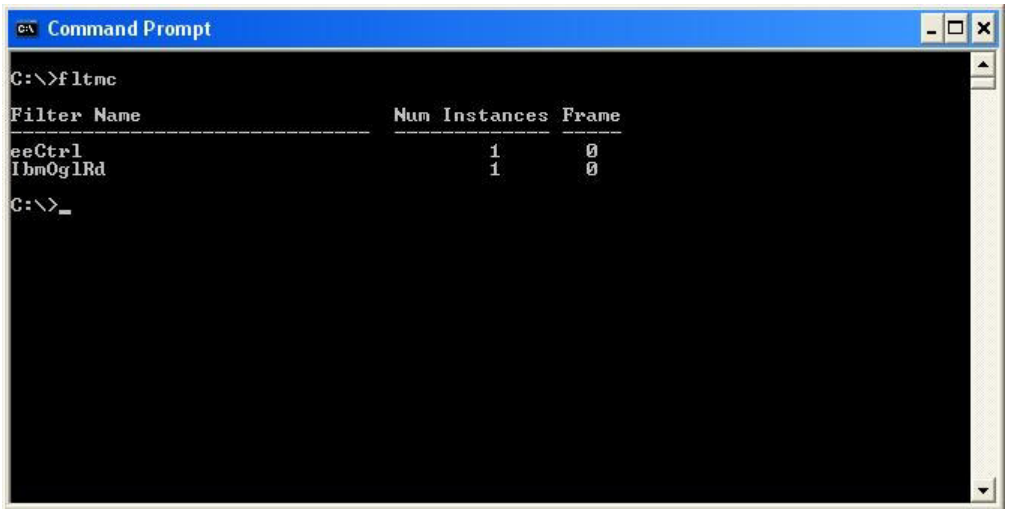

Figure 1. File System Minifilter driver - output (Deep Computing Visualization installed)

If the administrator needs to disable this driver it can be done by running *dcv_disable.exe* at the command prompt. To re-enable the driver run *dcv_enable.exe* at the command prompt.

Deep Computing Visualization Test Application

About this task

Included in the Deep Computing Visualization installation is a Deep Computing Visualization test application. This can be used to test RVN, SVN and CVN.

- **Windows** - To launch the Deep Computing Visualization test application: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > DCV Test App**.
- **Linux** - Enter **dcv_test** at a command line.

Example

Figure 2. Deep Computing Visualization Test Application

Verifying Installation

About this task

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

Verification of RVN Installation on Linux

About this task

To verify RVN installation on Linux, follow these steps:

1. Run the **rvn_console** command on the application host.
2. Click the **Enable** button
3. Run the Deep Computing Visualization test application on the application host - `dcv_test`. The application should start and a new tab should appear on the console.

4. Select and then clear the **Show on Server** check box to enable and disable the display on the application host. If the graphics are displayed and blanked out in accordance with the setting then the installation on the application host was successful.
5. Connect from an end station (on which Deep Computing Visualization including RealVNC VE Viewer 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 the end station, then the end station installation was also successful.

Verification of RVN on Windows

To verify RVN installation on Windows, follow these steps:

1. Start the RVN Console: Click **Start >All Programs > IBM > IBM Deep Computing Visualization > RVN Console** on the application host machine.
2. Click the **Enable** button.
3. Run the Deep Computing Visualization test application: Click **Start >All Programs > IBM > IBM Deep Computing Visualization > DCV Test App**. The application should start and a new tab (windcv_test.exe) should appear on the console.
4. Select and clear the Show on Server check box on the console to enable and disable the display on the application host. If the graphics are displayed and blanked out in accordance with the setting then the installation on the application host was successful.
5. Connect to the application host from an end station on which Deep Computing Visualization and RealVNC Viewer have been installed. If the 3-D graphics from the application are visible on the end station then the end station installation was successful.

Verification of SVN Installation on Linux

About this task

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). **For Example**, in order to run SVN using the default wall config file and sockets enter svn_enable - wall /opt/IBM/dcv/svn/examples/linux_sample_wall.config -transport SOCKET. See “The svn_enable command” on page 41 for additional information.
3. Run the Deep Computing Visualization test application, `dcv_test`. If the 3-D graphics from the application are displayed in a new window on the local system, then the installation was successful.

   Note: Check the parameters are correct in the console display, especially the location of the wall configuration file.

Verification of SVN on Windows

To verify SVN installation on Windows, follow these steps:

1. Log on to the machine

2. Ensure that the SVN Listener is running (It is started automatically by the installation program and added to the system Start Up menu). For more information, see "The SVN Listener" on page 39.
3. Run the SVN Console: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Console**.
4. Click the **Enable** button.
5. Run the Deep Computing Visualization test application by selecting **Start > All Programs > IBM > IBM Deep Computing Visualization > DCV Test App**.
6. If the application starts and a new window appears on the top left of the display showing the 3-D graphics from the application, then the installation was successful.

Note: Check the parameters are correct in the console display, especially the location of the wall configuration file.

Chapter 4. Configuring RVN

Before using RVN, you need to configure the application host machine and each end station.

Application hosts and end stations

RVN uses an application host to run OpenGL graphics applications and transmits the output to one or more end stations that connect to application host 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 running on the host machine.
- Sending user interaction with an 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 are displayed in an application window within the VNC Viewer window. You must install RealVNC VE to access RVN in VNC mode.
- In X11 Export mode, the graphic output is displayed in the application window on the exported display. X11 Export mode is available only on Linux systems.

A variety of configurations are possible:

- RVN in VNC mode (see Figure 3 on page 16)
- RVN in VNC mode with desktop isolation — Linux application hosts 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 with a single end station or in a collaborative configuration with multiple end stations that can join the community on the fly.

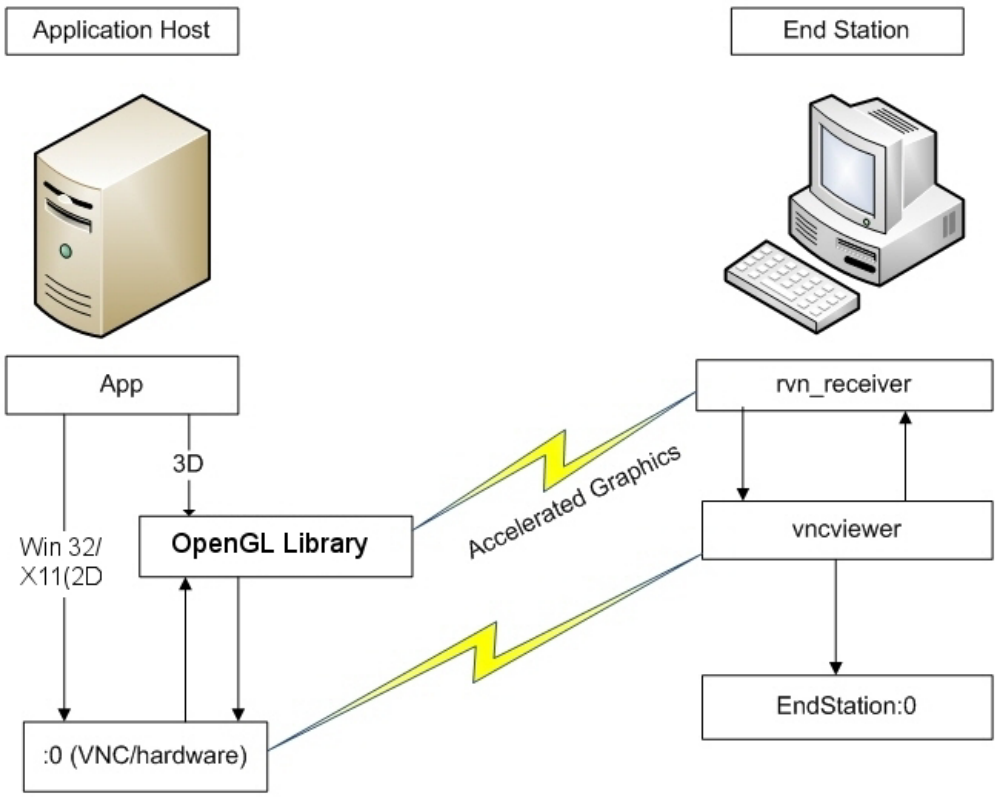

Figure 3. RVN in VNC mode

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 display 3-D data on 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.

Note:

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.

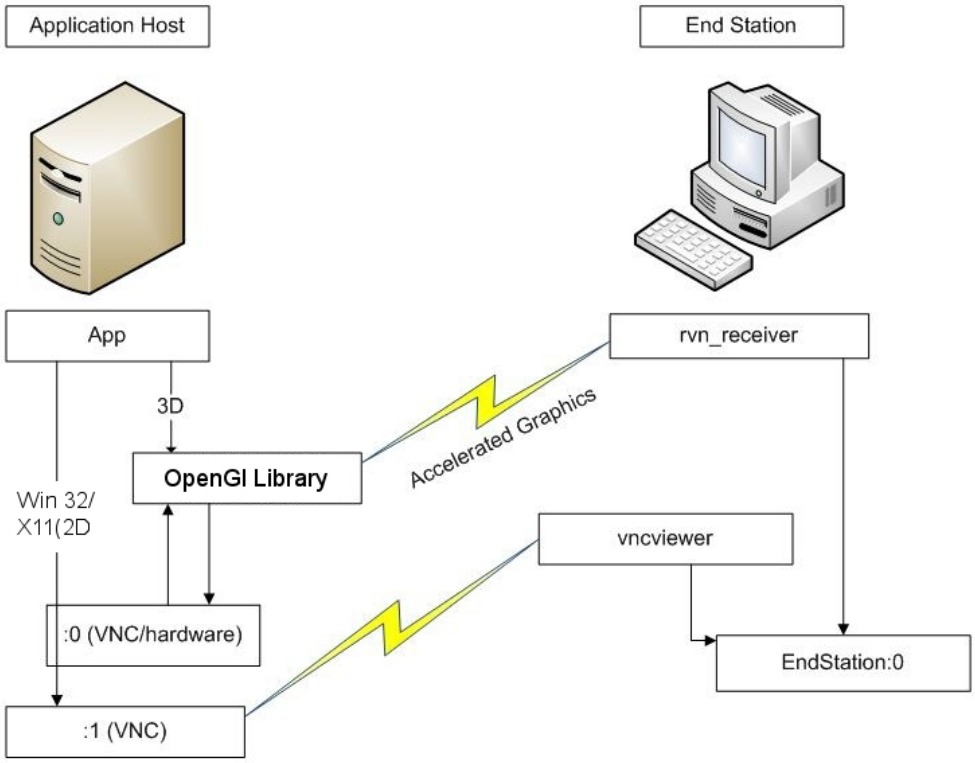

Figure 4. RVN in VNC mode with desktop isolation

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. All virtual sessions remain separate and independent of one-another

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.

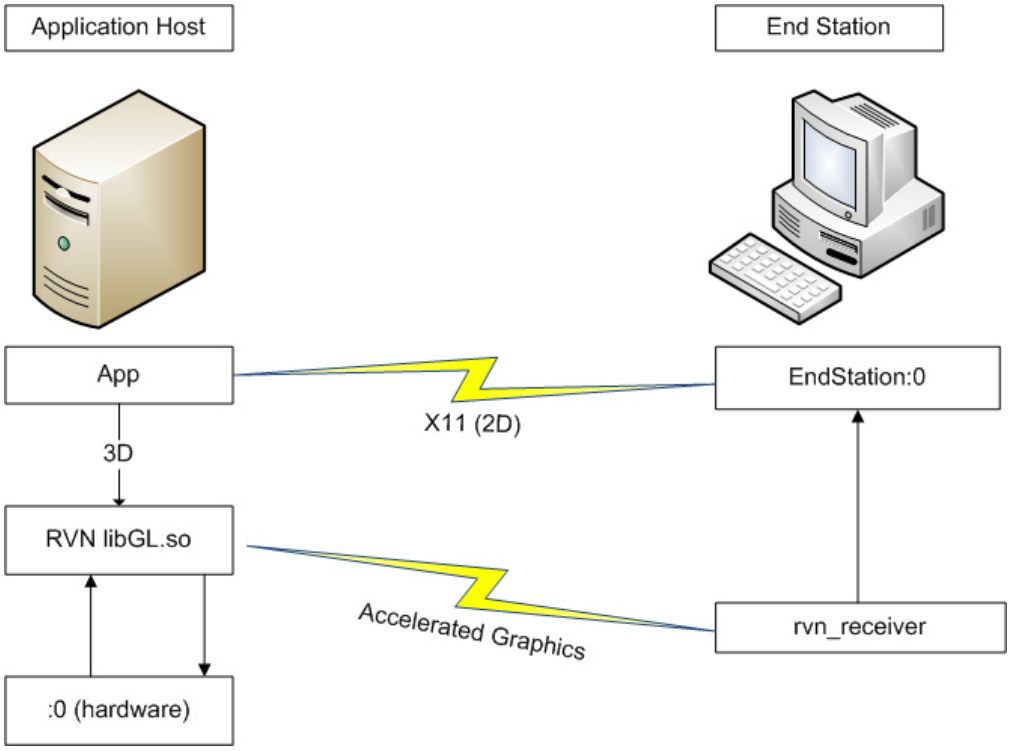

Figure 5. RVN in X11 Export mode

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.

Note:

1. The application host and end station must be running Linux.
2. Use the RVN Console(and not `rvn_enable`) to start your application — this ensures correct authentication via conference key and pass code.
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.
7. To run an application with RVN in X11 Export mode see "Running an application with RVN in X11 Export mode" on page 29.

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** through the RVN Console 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 the `rvn_coordinator` automatically manages the connection between the application host and connected end station.

The process by default 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.

Note:

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

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 5. 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 on both operating systems.

The RVN Console

The RVN Console allows you to enable and disable RVN on a desktop, either on the console or a virtual desktop (by invoking the underlying enable & disable command line tools in response to button clicks on the UI). The console allows you to specify the default settings and to save user defined named presets.

RVN Console - Main Window

Figure 6 on page 22 shows the main window of the RVN Console.

If Deep Computing Visualization is not already enabled when the RVN Console is opened, a dialogue box with the following comment will appear: *DCV is not enabled, Enable Yes or No*.

Click **Yes** to enable Deep Computing Visualization or **No** to end the process.

The RVN console contains a Settings tab plus a dynamic list of tabs specific to each application running on the desktop. These application tabs are created when an OpenGL application starts and allows the user to control the RVN-related runtime parameters in use for this application. The “Settings” tab instead contains the settings that will be used when the next OpenGL application will be executed. The RVN Console provides the following runtime controls:

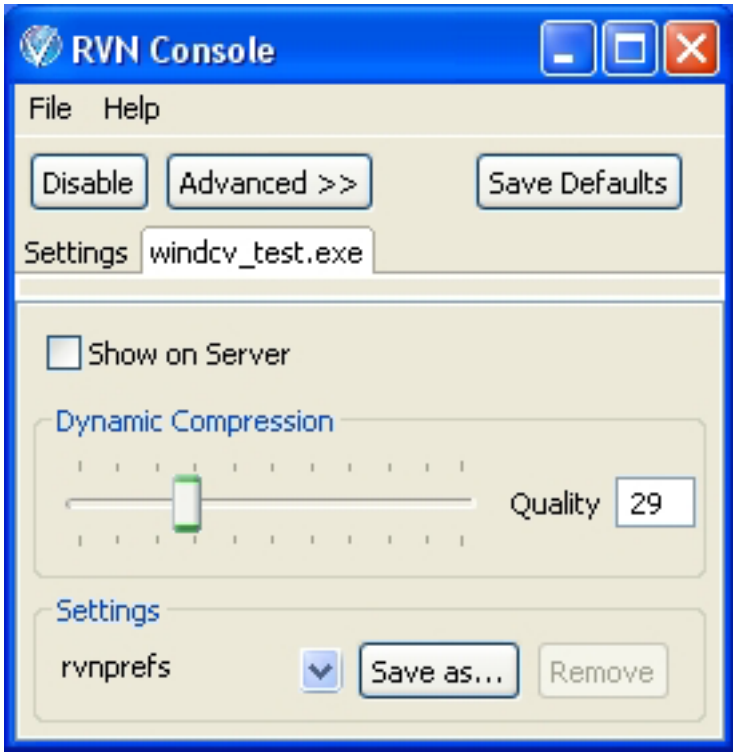

Figure 6. RVN Console main window

RVN Enable/ Disable

- **Enable** Clicking Enable allows you to enable RVN on a desktop, either on the console or a virtual desktop.
- **Disable** Clicking Disable allows you to disable RVN on a desktop

Save Defaults

- Click **Save Defaults** on the RVN Console, to save any changes that you have made.

Show on server

Displays the 3-D graphics on the application host in addition to sending it the connected end stations. If this check box is selected, RVN updates the application host screen directly. To change this setting, use the environment variable `RVN_HOST_SHOW_PIXELS=1` or load a saved profile. The default is **off**

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. Conversely a low value can increase performance at the expense of image quality.

Quality

Shows the current value for dynamic compression. Instead of using the Dynamic Compression slider, you can manually enter a value in the box, **Range: 0–100**

Settings

Performance settings can be accessed using the Settings drop down list box. This contains a list of predefined presets, plus **Save As** and **Remove** buttons.

Save As

This button allows the user to save the current settings into a new preset name, which will then be available in the drop down list box.

Remove

This button is used to delete an existing preset. System supplied presets can not be removed.

rvnprefs file

There are 3 default settings stored in the install path, these can not be edited on the console. These settings are:

low.rvn

medium.rvn

high.rvn

User Preferences

User preferences have a **.rvn** extension. Use the **Save As** button to save new preferences files, enter the name then click **Okay**. These files can be removed by selecting the file then clicking the **Remove** button. User preferences files are stored in:

Windows \documents and settings\usrname

Linux /usr/home

Advanced options

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

- **"Frame Sending Modes" on page 25**
- **"Quality Settings" on page 25**
- **"Long Distance Transport" on page 25**
- **"Show Partial Frames" on page 26**

To apply new settings, they must be saved by clicking **Save as**, entering a name then clicking **Ok**.

Note: Only the system administrator can edit 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 Console before the changes take effect.

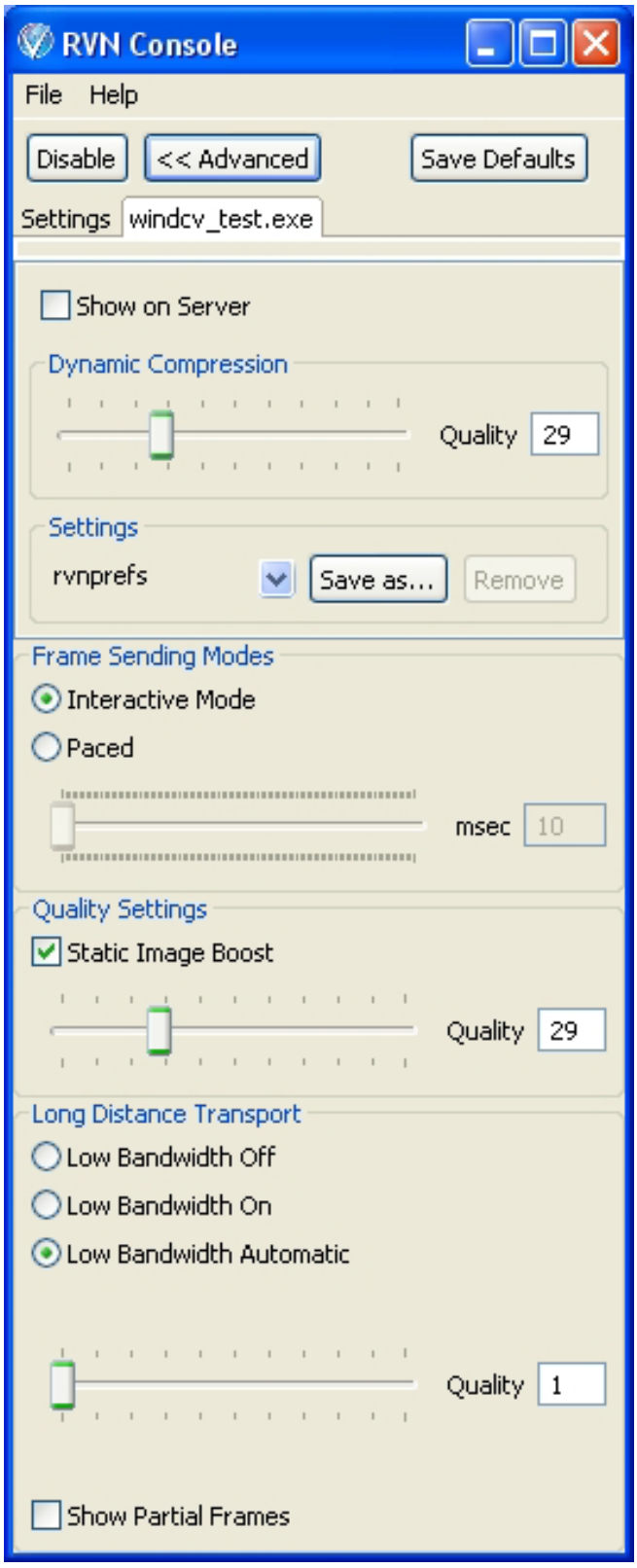

Figure 7. RVN Console with Advanced selected (Windows)

Frame Sending Modes

Interactive Mode

RVN compresses and sends the latest frame to each 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 may be the result of network load, **For Example**: 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.

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 the image stops changing. 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 when the image stabilizes, thereby enhancing the quality of the static image.

Note: Rapid flickering might occur when this check box is selected. This occurs when the application is rendering continuously but slowly enough that RVN has enough free bandwidth to interleave low quality and high quality images. To avoid this behavior, clear this check box. In this mode, the client firewall must be disabled or at least it must allow UDP network packets coming from the host machine. Failure to do so may result in flickering on the end station.

Long Distance Transport

Low bandwidth On

Low bandwidth optimizations are enabled.

Low bandwidth Off

Low bandwidth optimizations are disabled.

Low bandwidth Automatic

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

Image quality slider

Use the slider to adjust the image quality setting. You can also enter a number from 1 to 100 in the quality box where 100 is for the highest image quality.

Note: This option is available if you select Low bandwidth Automatic or Low bandwidth On

Show Partial Frames

Select the check box to show partial frames

Using RVN from the command line

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

rvn_enable.exe(Windows)/ rvn_enable(Linux)
Ensures that applications subsequently started on the X Server can use RVN.

rvn_disable.exe(Windows)/ rvn_disable(Linux)
Prevents subsequently started applications on the X Server from using RVN.

winrvn_console.exe(Windows)/ rvn_console(Linux)
Starts the RVN Console.

vncviewer.exe(Windows)/ rvn_viewer(Linux)
Starts a VNC viewer on an end station.

rvn_receiver.exe(Windows)/ rvn_receiver(Linux)
Receives data in X11 Export mode.

rvn_query.exe(Windows)/ rvn_query(Linux)
Displays a message indicating if RVN is enabled.

Note: If Deep Computing Visualization is not enabled on the system, run the `dcv_enable` command(administrator-level). (This is required only if you have disabled the software using, for example, the `dcv_disable` command or reinstalled your graphics drivers.)

rvn_enable.exe(Windows)/ rvn_enable(Linux)

`rvn_enable.exe(Windows)/ rvn_enable(Linux)` starts RVN on the specified display.

Linux The command uses the current DISPLAY environment variable or gets the display name from the command line option `–display` *dpyname*. The format of the command is:

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

rvn_disable.exe(Windows)/ rvn_disable(Linux)

rvn_disable.exe(Windows)/ rvn_disable(Linux) disables RVN functionality. OpenGL applications that are started after this command is run do not use RVN functionality.

Linux This command uses the current DISPLAY environment variable or gets the display name as a command line option `–display` *dpyname*. The format of the command is:

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

winrvn_console.exe(Windows)/ rvn_console(Linux)

`winrvn_console.exe(Windows)/ rvn_console(Linux)` starts the RVN Console. When the Console is started, any running OpenGL applications will connect to the Console(after a brief delay).

This is the syntax of the rvn_console command:

```
rvn_console
```

Note:

1. **Linux** The rvn_console command replaces rvn_dashboard.
2. There are no command line options.
3. For a description of the related environment variables, see "RVN Console" on page 61.
4. **Windows** The console must be started before the OpenGL application that has to be use remotely.

rvn_receiver.exe(Windows)/ rvn_receiver(Linux)

rvn_receiver.exe(Windows)/ rvn_receiver(Linux) is used in X11 Export mode to receive the pixel data.

Note:

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."

Command-line options for rvn_receiver

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

<*host*> The name of the host running the OpenGL application that is remotely accessed.

<*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 the conference participant before the conference starts.

vncviewer.exe(Windows)/ rvn_viewer(Linux)

Windows The system path is modified such that invoking the normal RealVNC Viewer will automatically activate RVN receiver functionality.

To start the vncviewer on Windows, click **Start > All Programs > Real VNC> Run VNC Viewer**.

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

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 at http://www.realvnc.com/products/visualization.

Running an application with RVN on Linux

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

- "Running an application with RVN in VNC mode"
- "Running an application with RVN in VNC desktop isolation mode" on page 29
- "Running an application with RVN in X11 Export mode" on page 29

Running an application with RVN in VNC mode

About this task

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

On the application host

1. Start the RVN Console.
2. On the RVN Console click the **Enable** button if RVN is not already enabled.
3. Start the OpenGL application. The application starts and a new tab will appear on the RVN console to control that application.
4. For information on adjusting settings on the RVN Console see "The RVN Console" on page 21

On each end station

1. Run **rvn_viewer**.
2. Connect to the application host.
3. Interact with the application, set various parameters in the console to improve graphics performance.

Running an application with RVN in VNC desktop isolation mode
About this task

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>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. Run the RVN Console.
3. Enable RVN using the **Enable** button.
4. Connect to the application host, display <N>, then launch your OpenGL application.
5. Use the tab on the console to control the runtime parameters for the application.

On subsequent end stations

1. Start **VNC Viewer**.
2. Connect to the application host, specifying display number `<N>`.

Running an application with RVN in X11 Export mode
About this task

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. **rvn_coordinator&**
2. **export RVN_USE_VNC=0**
3. **export RVN_CONFERENCE_ID=bar**
4. **export RVN_CONFERENCE_KEY=foo**
5. **export DISPLAY=<display>:0.0**
6. **rvn_enable**
7. **rvn_console&**
8. Start your OpenGL application.

   Note: All of these steps should be run on the same terminal window.

Note:

1. The application host and end station must be running Linux.
2. Use the RVN Console(and not `rvn_enable`) to start your application — this ensures correct authentication.
3. To change runtime settings, use the RVN Console as per VNC modes.

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

About this task

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

On the application host

1. Ensure the VNC VE Server is running on your machine
2. Start the RVN Console: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > RVN Console**.
3. On the RVN Console click the **Enable** button if required.
4. Start you OpenGL application. The application starts and a new tab will appear on the RVN console.
5. For information on adjusting settings on the RVN Console see "The RVN Console" on page 21

On each end station

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

Chapter 6. Configuring SVN

Before using SVN, you need to configure the application host and rendering servers. Figure 8 shows an example of SVN deployment.

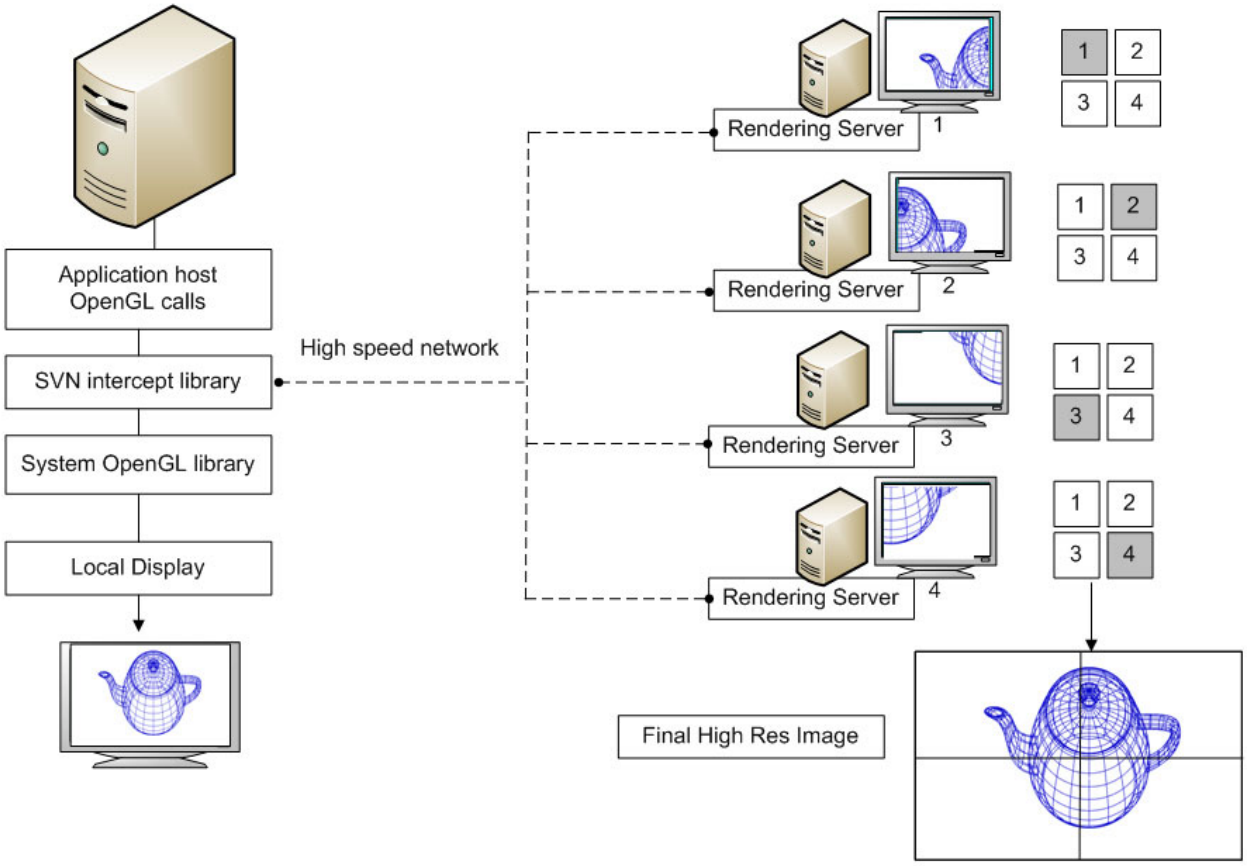

Figure 8. Deployment of Scalable Visual Networking

Display wall configuration

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 9 on page 32).

```
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
```

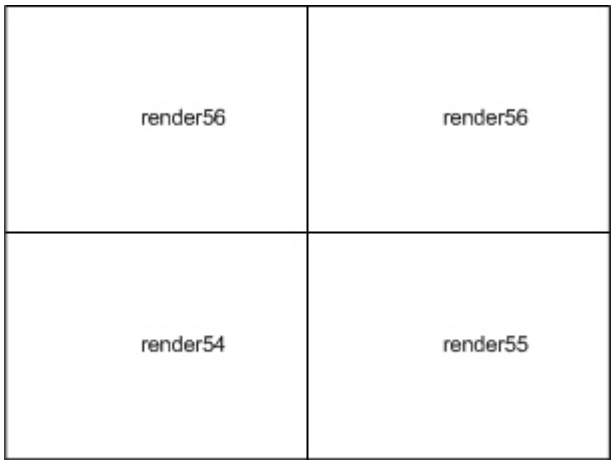

Figure 9. Virtual display with 2 x 2 tile array

Tiling is the term used to describe the configuration of a display wall and is used to indicate which parts of the overall image an individual machine is responsible for. This description is defined in a *wall configuration* file that contains an entry for each rendering server.

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 each 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.

  Note: For Windows Machines, use `:0.0` in place of `:d.s`
- `vwidth` and `vheight` are the overall pixel 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 in pixels.
- `width` and `height` are the total width and height for the tile assigned to this server.

Note:

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 33.
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:
   - In the SVN Console, use **Browse** to specify the path in the display. Alternatively use the -wall command line param and the svn_enable cmd
4. All values are expressed in pixels at the pixel pitch of each display device.
5. The display areas specified for rendering servers can overlap, and the pixel pitch of each tile can be different.

6. Blank lines in the wall configuration file, or lines that start with the # symbol, are ignored.

Sample wall configuration file

The installation procedure creates a sample wall configuration file in the following locations:

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

SVN transport options

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

Note: SVN Windows only supports TCP Sockets.

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. **For Example**, identical X server patch levels, kernel levels, video drivers.

Xinetd config

The default communication mechanism use to launch the SVN rendering servers when an SVN-enabled application starts relies on xinetd. The configuration file is updated to launch the SVN rendering server by installing a listener service which manages the handshake and process creation request from the application. The default listen port is **2007**

You should ensure that appropriate permissions are set to allow the user the xinetd listener service 'runs as' has access to the display group. See "NVIDIA display problems" on page 59

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. On Linux select **System Settings/ Login Screen/ Security**. Uncheck **Always disallow TCP connections to X server**.

Disabling SDP

If the SDP module is loaded by the kernel but no SDP stack/hardware is available, it may be necessary to specify `export SVN_DISABLE_SDP=1` before launching an application

Windows Considerations for SVN

- Ensure Windows Firewall allows listening on port 2007.
- Allow access through the firewall for any client app.

General Considerations for SVN

- All machines should be on the same network.
- If there are multiple network adapters per machine it may be necessary to specify the **Local Address** parameter in the Advanced section of the SVN Console.
- Stereoscopic configurations can be achieved using SVN even for applications which do not support it natively through the use of overloads. See Chapter 9, “Advanced users,” on page 47.
- Although SVN supports ’Windows to Linux’ and Linux to Windows’ interoperability mode, mixing operating systems within the rendering servers is not supported. Only the application host may vary in this case.
- Due to the operating system differences, application compatibility may be reduced when operating in mixed mode ’interoperability’ mode.

Chapter 7. Using Scalable Visual Networking

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

Running an application with SVN

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

1. Define the responsibilities of each of the rendering servers based on your configuration requirements. Store this information in the *wall.cfg* file. (For further information, see "Display wall configuration" on page 31.)

   Note: This step might already have been completed by the system administrator.
2. Start the SVN Console:
   - **Windows:** Click **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Console**.
   - **Linux** Run the **svn_console** command (Linux Only).
3. On the SVN Console click the **Enable** button or run the **svn_enable** command (Linux Only).
4. Click the **Browse** button then select the path to the wall.cfg file, if necessary.
5. Ensure that the SVN Listener/ xinetd is running on remote machines. See "The SVN Listener" on page 39
6. Start your OpenGL application by double-clicking its icon or issuing a shell command. The graphics output is displayed on the wall.

The SVN Console

The SVN Console is a graphical user interface used to enable and disable SVN on a desktop and to specify various configuration parameters.

SVN Console - Main Window

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

- "SVN Enable/ Disable" on page 36
- "Save Defaults" on page 36
- "Wall Configuration File" on page 36
- "Clientless Rendering" on page 36
- "Show Pointer" on page 36
- "Window Selector" on page 37
- "Frame Lock Support" on page 37

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

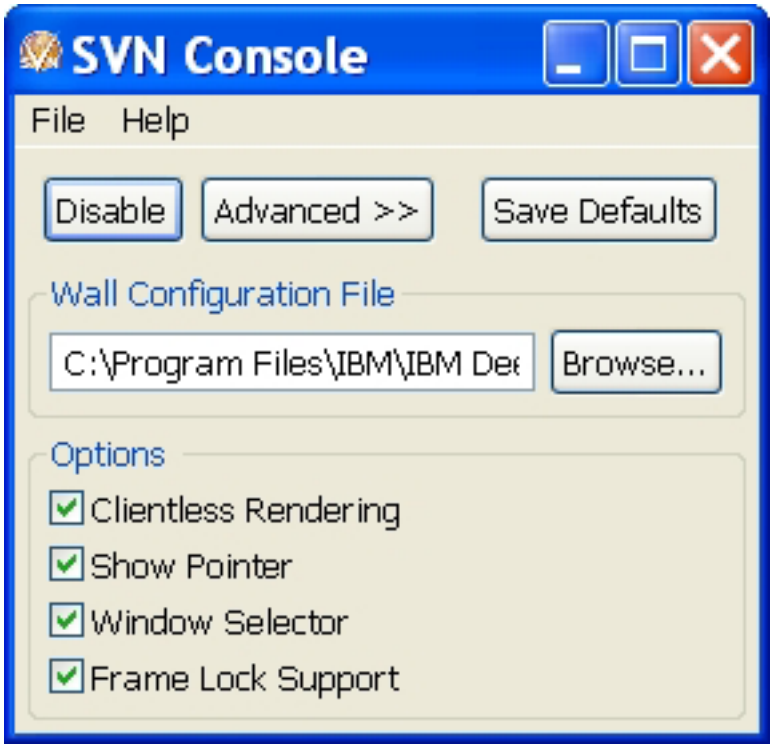

Figure 10. SVN Console

SVN Enable/ Disable

- **Enable** Clicking Enable enables the use of SVN for applications that are subsequently started on the application host.
- **Disable** Clicking Disable prevents the use of SVN by applications that are started after the button is clicked.

Save Defaults

- Click **Save Defaults** on the SVN Console, to save any changes that you have made.

Wall Configuration File

The path to the `wall.cfg` file.

Clientless Rendering

Turn off rendering on client, the display will be directed to the SVN wall only.

Show Pointer

Activating Show Pointer allows you to control a pointer that is displayed on the wall. Moving the mouse over the 3D window will update the pointer position at the wall

Window Selector

This option specifies whether a selector dialog will be displayed when you start your application. See "The SVN Window Selector" on page 40. You can use the SVN Window Selector to choose which 3–D Window to display on the wall.

Frame Lock Support

Frame locking 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. Use this option to enable framelock support in SVN. See "NVidia framelock Support" on page 40.

SVN Console Advanced Options

Clicking **Advanced** allows you to specify the following :

- "Synchronization" on page 39
- "Socket Options" on page 39
- "MPI Options (Linux only)" on page 39

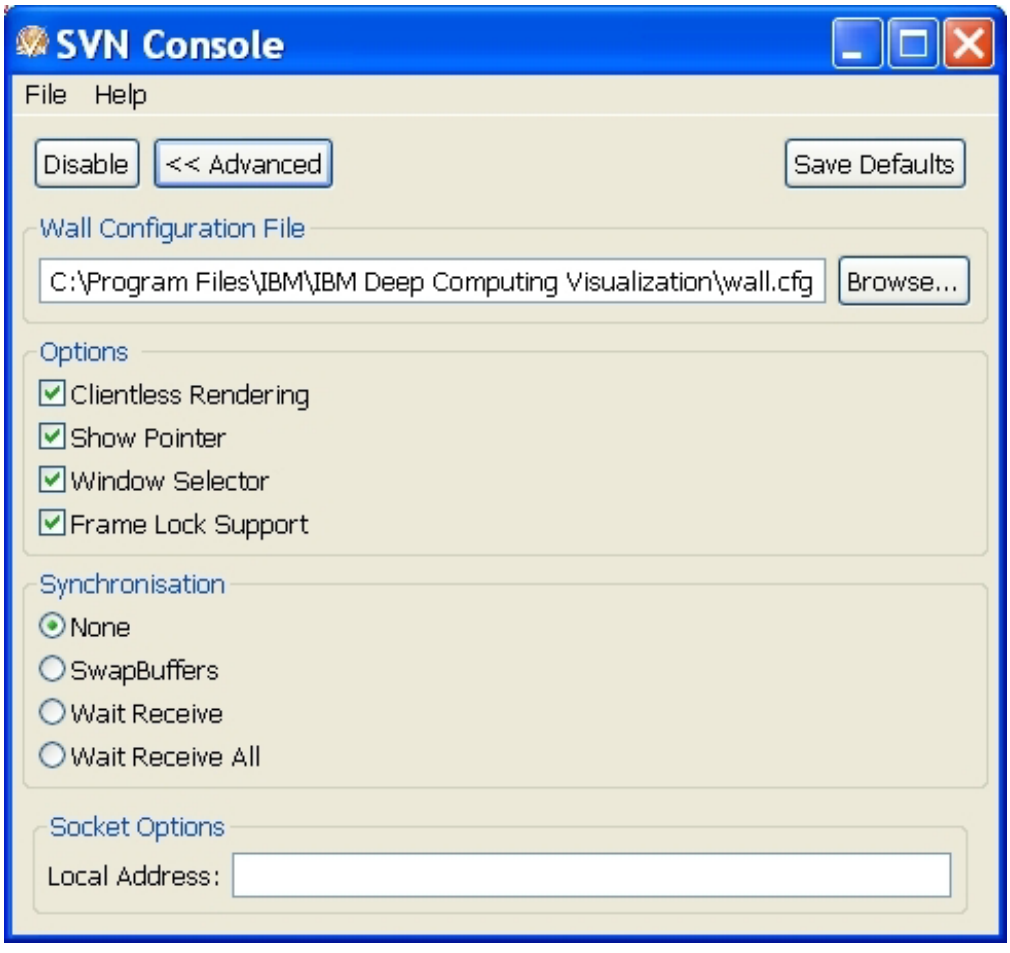

Figure 11. SVN Console with Advanced Options Windows Selected

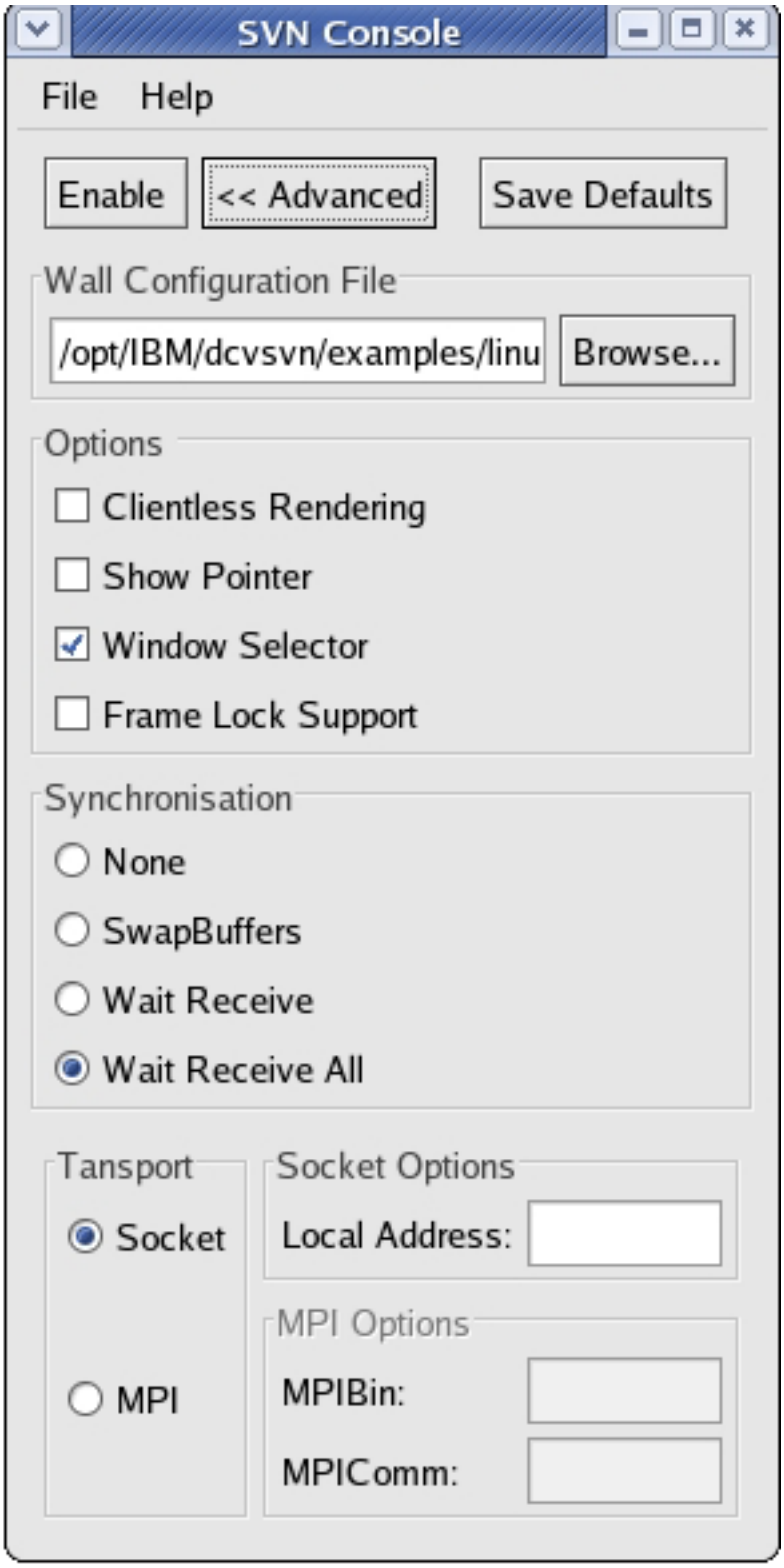

Figure 12. SVN Console with Advanced Options Linux Selected

Synchronization

- **None** Rendering servers will run as fast as possible but may fall quickly out of sync with one another.
- **Swap Buffers** Rendering servers receive new data once per frame.
- **Wait Receive** Each rendering server acknowledges receipt.
- **Wait Receive All** All rendering servers are updated together in lock-step.

Socket Options

- **Local Address:** The network address the rendering servers use to make connections back to the client applications. Use this when there are multiple adapters per machine.

MPI Options (Linux only)

If MPI transport is selected then MPI group is enabled

- **mpibin** (Linux only).
- **mpicomm** (Linux only)

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 13. The SVN Window Selector

Windows

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

Linux

- The Listener is used to start the rendering servers and is launched from the xinetd service on demand. By default this service is configured with minimal access. You may wish to improve this. See “NVIDIA display problems” on page 59

The SVN Window Selector

When an application creates multiple 3-D Windows or multiple 3-D applications are running concurrently. You can use the SVN Window Selector to choose which 3-D window to display on the wall.

Figure 14. 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.

Note: Windows Only: The cursor changes to a crosshair when you hover over a suitable 3-D window.

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).

Note:

1. On Linux, there is a single SVN Window Selector per desktop. On Windows, there is an SVN Window Selector per application, and each selector only has an effect on windows from the associated application.
2. The SVN Window Selector starts automatically based on the settings in the SVN Console or supplied in the command line.

NVidia framelock Support

Framelocking is the process of synchronizing multiple output displays from a master system to the other systems in the network, ensuring that the displays are synchronized with one other.

Note: SVN does not require that the application host be part of the framelock group, but does support such a configuration.

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 and connect them using the appropriate cabling.
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.” Alternatively use the SVN Console to enable framelock.

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.

Note: The selector can be used to reengage framelock. If framelock support is enabled but the display loses framelock, you can attempt to reengage it by using SVN Selector to reselect the relevant window

Using SVN with Linux Shell

Note: The SVN Console can be used to change settings, use the **svn_console** command to open the SVN Console. See “The SVN Console” on page 35.

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

svn_enable
: Ensures that applications that are subsequently started on the display 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` adminstrator-level command. (This is required only if you have disabled the software using, for example, the `dcv_disable` command.)

Note: The `dcv_disable` command on the rendering servers doesn’t effect the rendering servers operations whilst using SVN.

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*} **Linux**
Contains the display name on which you wish to enable SVN functionality.

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 SVN enabled.

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

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

Note:

1. If the command that is used to start MPI is not in your 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*} **Linux**
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 40). 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 but may fall quickly out of sync with one another.

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 in lock-step.

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

-transport { OPENMPI | SOCKET } (default) Linux
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 and specifies the configuration of the display wall. (For more information, see "Display wall configuration" on page 31.)

For 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.

SVN Interoperability

SVN was designed to display high resolution graphics on a rendering cluster where the operating system for the cluster matches the operating system on the host machine. However another valid configuration allows for a mixed mode environment where the application host operating system is different to that on the rendering cluster, **for example** Application Host - Windows, Rendering Cluster - Linux .

Note:

1. Not all application are guaranteed to work with this configuration, especially if they use OGL capabilities specific to the host OS/driver.
2. 64 to 32 bit interoperability is not supported, however 64 bit installation also includes the 32 bit install. This allows 32 but applications to be run on a 64 bit setup.
3. All machines in the rendering cluster should be homogeneous, **for example** all Linux or all Windows.

Chapter 8. Collaborative Visual Networking (CVN)

Both the SVN and RVN functions of Deep Computing Visualization can operate simultaneously in a joint CVN session. See Figure 15 CVN Interoperability Basic Diagram.

CVN can run in the following modes:

- RVN rendering on the application host
- RVN application display on a dedicated rendering server **(Linux Only)**

Note: For more detailed information on RVN and SVN, please see:

- Chapter 4, "Configuring RVN," on page 15
- "Using RVN from the command line" on page 26
- Chapter 6, "Configuring SVN," on page 31
- Chapter 7, "Using Scalable Visual Networking," on page 35

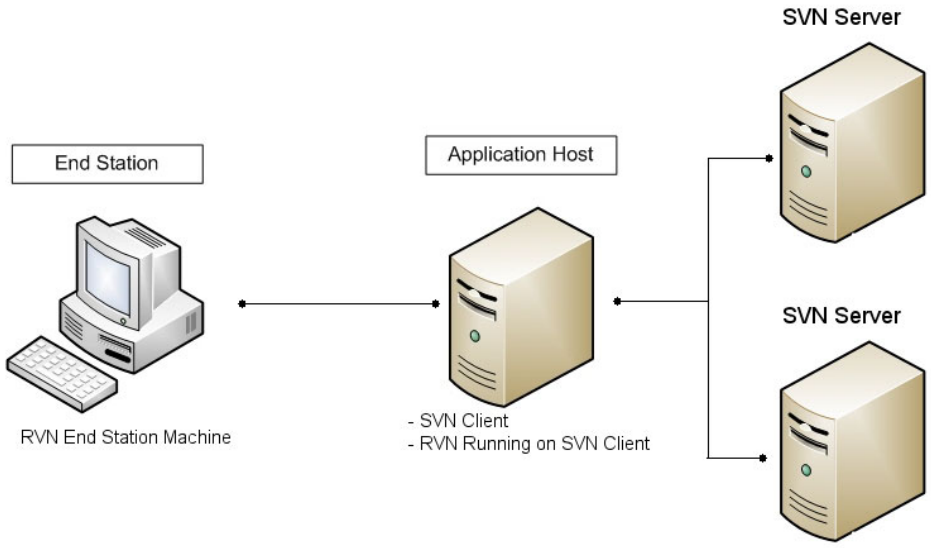

Figure 15. CVN Interoperability basic diagram

RVN rendering on the application host

About this task

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 or use the SVN Console.
2. Enable RVN: Run the **rvn_enable** command or use the RVN Console.
3. Launch your OpenGL application. The application displays on the SVN wall.
4. Connect to the application host from an RVN end station.
5. Start the RVN Console to control the remote session parameters.

Example

For Example (Linux)

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 (Linux only)

About this task

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 SVN application host
3. Enable RVN: Run the **rvn_enable** command on the dedicated rendering server.
4. Launch the Deep Computing Visualization test application, **dcv_test**. The application displays on the SVN wall.
5. Connect to the rendering server from a standard RVN end station.
6. Start the RVN Console to control the remote session parameters.

Example

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 programmers who wish to enhance or otherwise change the behavior of OpenGl applications using the capabilities of Deep Computing Visualization.

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 CVN)

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. Some 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.

Likewise programs that make us of shaders, and store the client window size, the window may not work correctly.

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

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.

Note:

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
   - **Linux** */opt/IBM/dcv/svn/examples/overload* directory/
   - **Windows** c:Program Files\IBM\IBM Deep Computing Visualization\ examples\overload\

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.

Note: It is likely that overloads will need to be recompiled against each release of Deep Computing Visualization, as more functionality is included and the numbering and ordering of OpenGL instructions changes.

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, we explicitly overload `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 is used to call the current SVN implementation of the `glColor3f` function.
- 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) version 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, where the reference is 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

- **Linux** `<install dir>/examples/include/wire.h`
- **Windows** `<install dir>\examples\include\wire.h`

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 existing SVN version of the 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 SVN 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, **Linux** *`invert_colors.so`* or **Windows** *`invert_colors.dll`*).

Note: You can use the sample *Makefile* or Visual Studio Project file provided with the SVN code.

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

   Note:

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

   b. The format for each entry has the form:

      - **Linux:** *<absolute_path>/invert_colors.so*
      - **Windows:** *<absolute_path>/invert_colors.dll*

      .

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`. 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" on page 48 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 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 rendering server, the return value of `DVGetDisplayName()` points to a string containing the name of the 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 app host, 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 *<install directory>/include/wire.h*. The `wire.h` file needs to be included in the overload source code file.

Overload example: inverting colors on Rendering Servers only

To invert application colors on the SVN Rendering 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;
}
```

Overload Example - Stereo Overload

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 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 application host can have different behavior:
  - 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, **Linux** *stereo_overload.so* or **Windows** *stereo_overload.dll*).

     Note: You can use the sample *Makefile*/ Visual Studio Project file 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:
       - **Linux** *<path>/stereo_overload.so*.
       - **Windows** *<path>/stereo_overload.dll*.
  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 application host 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 you OpenGL application and feed the left eye and right eye displays to appropriate stereo projection hardware.

Note:

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 applied in the reverse order listed. 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:

- **Linux**

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

- **Windows**

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

Note:

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:

- **Linux**

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

- **Windows**

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

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 16 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(...)` does 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.

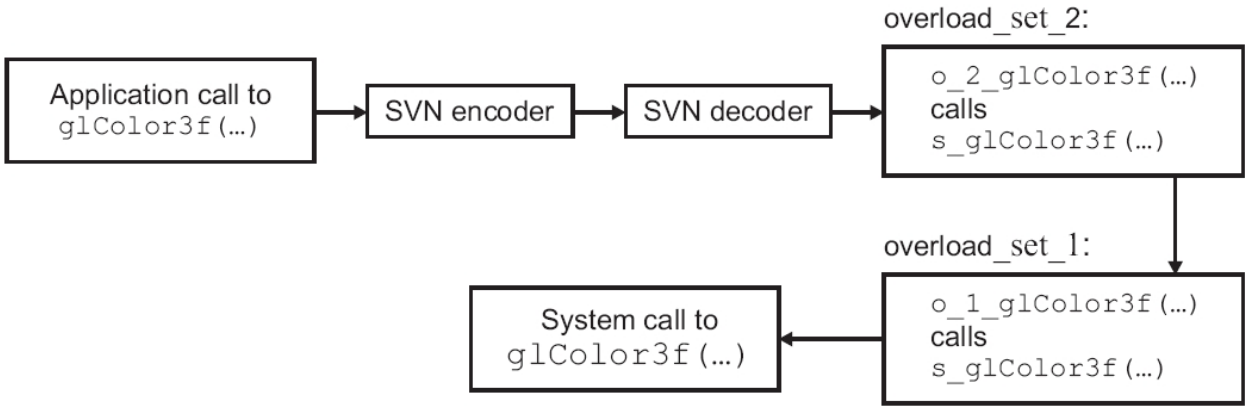

Figure 16. Multiple overload sets

Chapter 10. Diagnostics

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

Note:

1. Appendix B, "Error Messages," on page 67 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 x.

RVN problems

About this task

Linux

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. Select **System Settings/ Login Screen/ Security**, uncheck **Always disallow TCP connections to X server**. See also "GDM security" on page 34.
2. If you cannot get the application to open the display, follow local problem-reporting procedures.

Windows

If the application host reports that it cannot export the display,

1. Run an application directly on the application host without endstations. Verify that the application responds correctly to show on server being enabled/disabled
2. Connect a single endstation, verify 3d graphics are shown and respond to changes in quality settings on the console

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 5, "Using

Remote Visual Networking," on page 21). 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. 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.
4. Run the installation verification procedure. See "Verifying Installation" on page 11.

SVN problems

About this task

Linux

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 rendering 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.
- On each rendering server a log file is created for each server run containing the stdout for that server run. It may give insight into the cause of a problem and is located at **/tmp/svnsvrlog.txt**
  - 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 Window Selector and select the display window several times. Sometimes an application will create temporary windows that are never destroyed, use the Cycle Window option to loop through all OpenGL windows that the application has created until the correct rendering appears.

   Note: It may be necessary to force the application window to refresh for rendering to appear on the wall.
2. Try running the `dcv_test` application to confirm system configuration.
3. Check the contents of the wall configuration file, to ensure that the correct entries are specified for the desired wall.
4. Reboot and try the operation again.
5. 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 (direct) rendering is being used. Check also that a high-performance network is being used by both the application host and the rendering servers.

Confirm that the rendering servers have the correct permissions to access the graphics hardware.

Note: For security reasons, the rendering servers are installed and run as user **NOBODY** by default. This will result in **Indirect Rendering** unless sufficient permissions are granted to users or the user the server runs as is changed (see "NVIDIA display problems" on page 59)

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. Examine the svn server log (in **/tmp/svnsvrlog.txt**) on each rendering node for information.
4. 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 34.
- If you cannot get the application to open the display, follow local problem-reporting procedures.

If using RSH/SSH

- Review the appropriate `RSH` or `SSH` documentation and ensure that the access for `RSH` or `SSH` has been set up correctly.
- Make sure that the pass-phrases or public keys have been supplied to the SSH agents on the application host and on all of the rendering servers, as described in the `SSH` documentation.
- Ensure all machines are resolvable to IP adresses and that each machine can see all the others
- It is recommended that all SVN machines are on a trusted network, allowing firewalls to be disabled for debugging purposes.
- 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.

Windows

When using the SVN Console to alter SVN settings, make sure to save any changes and re-enable SVN when done.

With SVN is enabled, the launching of any OpenGL application should launch SVN on all rendering servers also.

- If an application fails to start on Windows, the SVN Listener might have been disabled on one or more of the rendering servers. Try to run the application and command console to view any error messages. To run the Listener: Click **Start > All Programs > IBM > IBM Deep Computing Visualization > SVN Listener**.

- Try running the `dcv_test` application to confirm the system is configured correctly. Click **Start > All Programs > IBM > IBM Deep Computing Visualization > DCV Test App**.
- If you are still experiencing problems follow local problem-reporting procedures.

Windows and **Linux**

- Ensure you are not attempting to run a 64-bit application to a 32-bit rendering server, such operation is not supported. Only 32-bit applications may be run to a 32-bit server, 64-bit applications require a matching 64-bit server

If you receive a `permission denied` message:

- If the machines have multiple network adapters, ensure the 'local address' is specified correctly to allow the rendering servers to connect back to the application over correct adapter

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

X Server display problems

About this task

Linux

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 the X Server.
- xdpyinfo
  - This display utility provides information for X, including whether accelerated 3D graphics are enabled..
- xhost
  - Shows information about which machines are allowed to display on the local X Server.

Display problems

About this task

Linux

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 ensure that the SVN server processes have read/write access to /dev/nvidia, by ensuring the user value specified in/etc/xinetd.d/svnlisten references a valid user with access to these devices.

Flickering (RVN)

Example

Windows and **Linux**

Your display might appear to flicker because the application is rendering continuously but with sufficient time for RVN to produce and send a quality update after each frame. To resolve this, clear the Static Image Boost box.

Flickering (SVN)

Example

Linux

If you experience an application flickering on the rendering servers, but running correctly on the application host then set the following environment variable:

export SVN_FLICKER_REDUCTION=1

This may eliminate the flickering, which is due to certain combinations of video driver and display /X server setups.

NVIDIA display problems

About this task

Linux

Ensure the user that is used to launch the application and the SVN rendering servers has read/write permission to /dev/nvidia*. This can be temporarily achieved if root executes `chmod 666 /dev/nvidia*`, but this will reset after the next reboot. For a more permanent solution edit the file **/etc/xinetd.d/svnlisten** and change the USER that the SVN Listener is launched under to one with graphics permissions (any regular user account will generally suffice).

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 4. If you reinstall display drivers, you will need to run the `dcv_enable` adminstrator-level 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).

Environmental variables for the RVN Application Host

RVN Console

In addition to the options listed in "The RVN Console" on page 21, you can also define the following environment variables to control **rvn_console** processing.

RVN_DASHBOARD_DISPLAY= :0.0 (Linux only)
Specifies the display on which to show the dashboard .

RVN_DUAL_COORDINATORS=[0|1]
Selecting 1 allows use of both minicoordinator and external coordinator process.

RVN_HOST_SHOW_PIXELS=:0.0
Specifies the display on which to show the dashboard

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.

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.

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.

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.

Note:

1. 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 and applies only to Jvl and Tiles compressors.

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 firewall must be disabled, or at the least accept UDP packets

RVN_UDP_QUALITY=[0...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.

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.

RVN_USE_SUPPLIED_CONFERENCE_ID=[0|1]
Specifying 0, use automatic conference ID. Specifying 1 uses the default conference ID.

RVN_VERBOSE=[0|1]
Selecting 1 displays diagnostic messages on console.

Environment variables for RVN in specific modes

In addition to the options listed in "The RVN Console" on page 21, you can also define the following environment variables to control **rvn_console** processing.

X11 Export mode

DISPLAY=*display*
Sets the display on which RVN is enabled. The default is :0.0

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

Note:

1. This information is needed only for X11 Export mode.
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.

Note:

1. This information is needed only for X11 Export mode.
2. This information must be sent to each conference participant before the start of the conference.

RVN_USE_VNC=[0,1]
VNC is set as default rather than SSC. For X11 export this must be set to **0**.

Other Settings

RVN_EXTERNAL_TRANSPORT=[0,1]
VNC is set as default rather than SSC. To change to SSC, set to **0**.

RVN_LOCAL_DISPLAY=*display* **(Linux)**
The default hardware-rendering adapter is :0.0. Use this environment variable to specify non-default hardware.

RVN_ORIGINAL_DISPLAY=*display* **(Linux)**
Used if RVN_DASHBOARD_DISPLAY is set in order to retrieve original display for socket code. Default :0.0.

RVN_PRESET= *filename*
Specifies the settings file that an rvn_enabled application will load on start up.

Mandatory settings for certain applications

RVN_COLOR_CONVERSION=[0|1]
In some cases (running specific applications with specific processors) a crash within a color conversion function used by the IPP JPEG codec can occur. For such cases, we provide an alternative implementation of the color conversion function. In order to use the non-IPP version of the color conversion function set this environment variable to **1**. In order to use the IPP version of the color conversion function set this environment variable to **0**.

RVN_UNIQIFY_CONFERENCE_ID=[1|0] (Linux)
(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.

Note:

1. This variable is relevant only for sessions that use X11 Export. It is not needed for sessions that use VNC.

Network Ports

RVN_COORDINATOR_PORT=*7200*
Specifies the TCP port on the application host used by `rvn_console`, `rvn_receiver`, and `rvn_coordinator` to establish an RVN conference session. This environment variable must have the same value for all three processes (`rvn_console`, `rvn_receiver`, and `rvn_coordinator`) participating in the RVN conference.

Note:

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

RVN_TOP_PORT=*7220 (or RVN_COORDINATOR_PORT+20)*
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.

RVN_UDP_START_PORT=*7400*
Sets the first port that can be used for UDP packets transfer.

RVN_MINI_START_PORT=*7300*
The first port that can be used by RVN minicoordinator.

RVN_MINI_MAX_PORT=*7399*
The last port that can be used by RVN minicoordinator.

RVN_PORT=*2000*
The port that the console is listening on. **(Windows Only)**

Codecs

RVN_CODEC1=*""*
Replaces full frame (TCP) encoder/codec (RVN_CODEC1 is also used for tiles-based (UDP) encoder/codec if the RVN_CODEC2 environmental variable is left unset). Set to the filename of one of the codec libraries in the codecs directory.

Note: This codec must support full frame compression (and optionally tiles-based compression).

RVN_CODEC2=*""*
Replaces tiles-based (UDP) encoder/codec. Set to the filename of one of the codec libraries in the codecs directory.

Note: This codec must support tiles based compression.

Environment variables for SVN

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

Note:

1. The environment variables listed in this section should be exported.
2. Where applicable, the underlined value is the default.

SVN_BANNER_FONT= *variable* **(Linux only)**
Name of font used in banner

SVN_BANNER_COLOR= *SkyBlue* **(Linux only)**
Name of color used in banner

SVN_BANNER_TIME = [#|4] (Linux only)
Number of secs to display banner

SVN_BIG_POINTS_SIZE = [#|2] (SVN Big Point o/l)
Size to use when displaying glPoint's.

SVN_DISABLE_SDP=[0|1]
If the SDP module is loaded by the kernel but no SDP stack/hardware is available, it may be necessary to specify export SVN_DISABLE_SDP=1 before launching an application. This disables use of SDP sockets, reverting back to TCP/IP sockets.

SVN_FLICKER_REDUCTION =[0 | 1] (Linux only)
If you experience an application flickering on the rendering servers, but running correctly on the application host then set

SVN_FLICKER_REDUCTION=1. This may eliminate the flickering, which is due to certain combinations of video driver and display /X server setups.

SVN_GECONFIG_FILE =*googleearth_shaders.cfg file-path* **(SVN Google Earth o/l)**
The configuration file for google earth overload

- **Windows** *C:\Program Files\IBM\IBM Deep Computing Visualisation\examples\overload\googleearth\googleearth_shaders.cfg*
- **Linux** */opt/IBM/dcv/svn/examples/overload/googleearth/googleearth_shaders.cfg*

SVN_KEEP_ASPECT_RATIO=[0 | 1]
For selection of the aspect ratio. Select 0 to use entire wall for display. Selecting 1, the aspect ratio on wall matches application.

SVN_OVERLOAD_FILE=*overload-file-path (no default)*
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(s). This sets the path to the main overload file and is only used if set. For more information about overloads, see "OpenGL overloads with SVN" on page 47.

SVN_PORT=*3001*
The port that the console is listening on.

SVN_REMOTE_PORT =*2007* **(Windows only)**
Port Windows listener listens on for connections from SVN clients

SVN_SOCKETS_PORT =*34567* **(sockets only)**
The start port for svn processes to connect with one another

SVN_SOCKET_START_CMD =*filepath* **(sockets only)**
This overrides the default socket start command

- **Windows** *<path>/svn_socket_start*
- **Linux***<path>\winsvn_socket_start.exe*

SVN_STEREO_FILE = *required (no default)*
The configuration file for stereo overload.

SVN_STEREO_MODE = [0 | 1]
Select 1 to auto-load the SVN Stereo overload.

SVN_STEREO_PROTUBERANCE = range[0 ..1] (SVN stereo o/l)
This sets the proturbance value for stereo projection.

SVN_STEREO_SEPARATION =*eye distance/ 30* **(SVN stereo o/l)**
This sets the separation of eye value for stereo projection

SVN_UNSCALED_BITMAPS = [0 | 1]

- With 0 selected, bitmaps are scaled to account for client vs wall size difference.
- With 1 selected, bitmaps sent over the wire aren't scaled.

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. (Linux only)

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. (Linux only)

Explanation You have not responded **Accept** 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

DCV RVN: Compression failed to initialize... exiting

Explanation Internal error.

User response Contact IBM Service.

DCV 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.

DCV 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.

DCV RVN: Tiles compression encoder returns error

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

User response Contact IBM Service.

DCV 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.

DCV 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.

DCV 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.

DCV 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.

DCV RVN: compression decoder returns an error

Explanation The RVN internal compression decoder returned an error.

User response Contact IBM Service.

DCV RVN: read call returns error

Explanation Attempting to read from socket returns error err.

User response Contact IBM Service.

DCV Bad magic number *number*

Explanation RVN received a corrupt packet.

User response Contact IBM Service.

DCV RVN: sender error writing handshake

Explanation Internal error.

User response Contact IBM Service.

DCV RVN: sender error reading handshake fd

Explanation Internal error.

User response Contact IBM Service.

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

Explanation Internal error.

User response Contact IBM Service.

DCV RVN: receiver read error

Explanation Internal error.

User response Contact IBM Service.

DCV RVN: receiver setup failure

Explanation Internal error.

User response Contact IBM Service.

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

Explanation The conference ID is not valid.

User response Use a different conference ID.

DCV RVN: Warning 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.

DCV RVN: Error 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: Connected downlevel DCV 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: 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 Console (Linux and Windows).

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

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 4.) 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: 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.

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.

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

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

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.

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

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.

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.

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*.

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.

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*.

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*.

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*.

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*.

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

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

R

S

T

U

V

W

X

Printed in USA