# DCV

# Administration Guide

Version **2016.0**

NICE

Last Update

September 21, 2017 (rev. 17066)

Latest Version

`https://www.nice-software.com/download/dcv`

# Contents

# Welcome

## About This Guide

This document describes the installation, configuration, and operation of NICE Desktop Cloud Visualization (DCV).

NICE DCV provides high-performance visualization of complex OpenGL and DirectX applications across low-bandwidth, high-latency networks to remote or distributed locations.

### Save this Document

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

### Who Should Read This Guide

This guide is intended for:

- system administrators who need to install and configure NICE DCV;
- users who want to remotely access their OpenGL and DirectX applications using NICE DCV.

# Learn About NICE Products

## World Wide Web

You can find the latest information about NICE DCV on its web site `https://www.nice-software.com/dcv`.

For more information about other NICE products and about the professional services provided by NICE you can refer to the company's web site `https://www.nice-software.com`.

Report problems accessing the aforementioned web sites to <`helpdesk@nice-software.com`>.

## NICE DCV Documentation

The latest NICE DCV documentation is available at `https://www.nice-software.com/download/dcv`.

# Get Technical Support

Contact NICE or your DCV reseller for technical support.

## NICE Support Contacts

Use one of the following to contact NICE technical support.

Email
<`helpdesk@nice-software.com`>

World Wide Web
`http://support.nice-software.com/`

Phone
+39 0141 901516

Mail
NICE Support
c/o NICE s.r.l.
Via Milliavacca, 9
14100 Asti
Italy

When contacting NICE, please include your company's full name.

# 1

# Introducing Desktop Cloud Visualization

DCV enhances the graphics functions of 3D applications on Linux and Microsoft Windows on both OpenGL and DirectX to display complex visual data across multiple, simultaneous distributed displays using low-bandwidth networks.

## Overview

NICE DCV is the remote 3D visualization technology that enables Technical Computing users to connect to OpenGL or DirectX applications running in a data center. Engineers and scientists are immediately empowered by taking full advantage of high-end graphics cards, fast I/O performance and large memory nodes hosted in "Public or Private 3D Cloud", rather than waiting for the next upgrade of the workstations. The DCV protocol adapts to heterogeneous networking infrastructures like LAN, WAN and VPN, to deal with bandwidth and latency constraints. NICE DCV multi-session capability allows IT Administrators to consolidate multiple users and/or application services into one node, or perform effective collaboration among multiple users.

Using NICE DCV, you can remotely work on 3D interactive applications, fully accelerated by high-end GPUs on workstations, blades or servers. No matter if you are accessing high-end OpenGL modeling applications or simple viewers, NICE DCV lets you connect quickly and securely from anywhere and experience high frame rates, even with low bandwidth standard Internet connections.

The product supports both Microsoft and Linux systems, enabling collaborative capabilities in heterogeneous environments. Moreover, it is perfectly integrated into NICE EnginFrame, leveraging 2D/3D capabilities over the Web, including the ability to share a session with other users for collaborative or support purposes.

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.

DCV 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.

When using DCV, the scene geometry and graphics state are rendered on a central server, and rendered images are sent to one or more remote displays.

NICE DCV includes the leading standard RealVNC Visualization Edition 4.6 to access and control your desktop applications wherever you are in the world, whenever you need to. VNC has a widespread user base, from individuals to the world's largest multi-national companies, utilizing the technology for a range of applications.

## Features

### Collaboration

Support for multiple, collaborative endstations. The set of endstations can be dynamic, with connections being made and others being dropped throughout the DCV session

### H.264-based encoding

Greatly reduces bandwidth consumption

### Exploits the latest NVIDIA Grid SDK technologies

Improves performances and reduces system load

Uses the NVIDIA H.264 hardware encoder (on Kepler and GRID cards)

### Full Desktop Remotization

Uses the high-performance NICE DCV protocol for the remotization of the full desktop (not only for the 3D windows as in previous versions)

### Support for NVIDIA vGPU technology

Simplifies the deployment of Windows VMs with full application support

### High Quality Updates

Support for high quality updates when network and processor conditions allow

### Image Only Transport

Transmission of final rendered image, rather than geometry and scene information, providing insulation and protection of proprietary customer information

### User Selectable Compression Levels

Ability to specify the compression level used to send the final image over the wire to the endstation

### Pluggable Compression Capability)

Pluggable compression/decompression (codec) framework, eventually allowing the image compression/decompression algorithms to be replaced

### Smart-card remotization

Seamlessly access the local smart card, using the standard PC/SC interface. Use smart cards for encrypting emails, signing documents and authenticating against remote systems

### Adaptive server-side resolution

Automatically adapt the server-side screen resolution to the size of the viewer window

### USB remotization (Preview)

Plug USB devices on the client side and use them on the remote desktop

## Typical Deployment

DCV uses an *application host* to run OpenGL or DirectX graphics applications and transmits the output to one or more *end stations* that connect to the application host over the network.

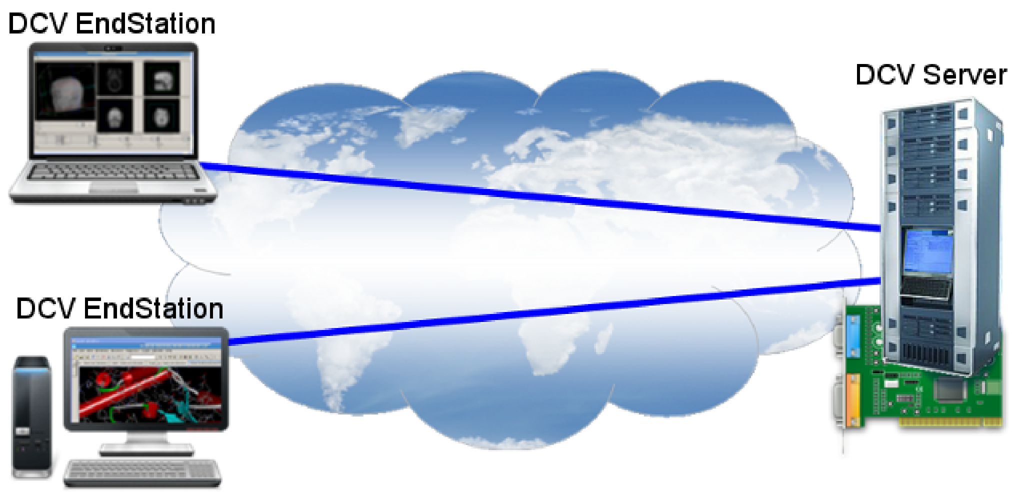


*Figure 1.1. NICE DCV*

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.

## Modes of Operation

DCV supports the following modes of operation:

*Table 1.1. Summary of DCV Modes of Operation*

| Technology | Hypervisors | Application compatibility | OS Support | |
|---|---|---|---|---|
| Bare metal or GPU pass-through | All | Maximum | Linux and Windows | Pros:<br>• Best performance<br>Limitations:<br>• One VM per GPU |
| NICE External Rendering Server | All | Limited | Windows | Pros:<br>• Best consolidation<br>• GPU sharing<br>• Dynamic GPU load balancing on multi-GPU hosts<br>• Support for GPU appliance mode<br>Limitations:<br>• No support for DirectX applications |
| NVIDIA vGPU | XenServer 6.2 SP1 | Excellent | Windows | Pros:<br>• Good performance<br>• GPU sharing<br>Limitations:<br>• Requires NVIDIA GRID cards and a specific hypervisor |

## NICE External Rendering Server

DCV application hosts can optionally be configured to delegate the actual 3D rendering to a separate host, the *rendering host*. In this case the OpenGL application runs on an application host which does not provide 3D hardware acceleration and delegates the OpenGL rendering to a rendering host equipped with one or more 3D accelerated graphic adapters.

This configuration enables *virtual machines* to act as application hosts even if the virtual hardware emulated by the hypervisor does not provide OpenGL acceleration.

The rendering host:

- receives OpenGL commands from the applications running on the application servers;
- sends the 3D image updates (in the form of pixel data) to each connected end station.

For more information on the platforms on which this configuration is supported please refer to Chapter 2, *Prerequisites*.

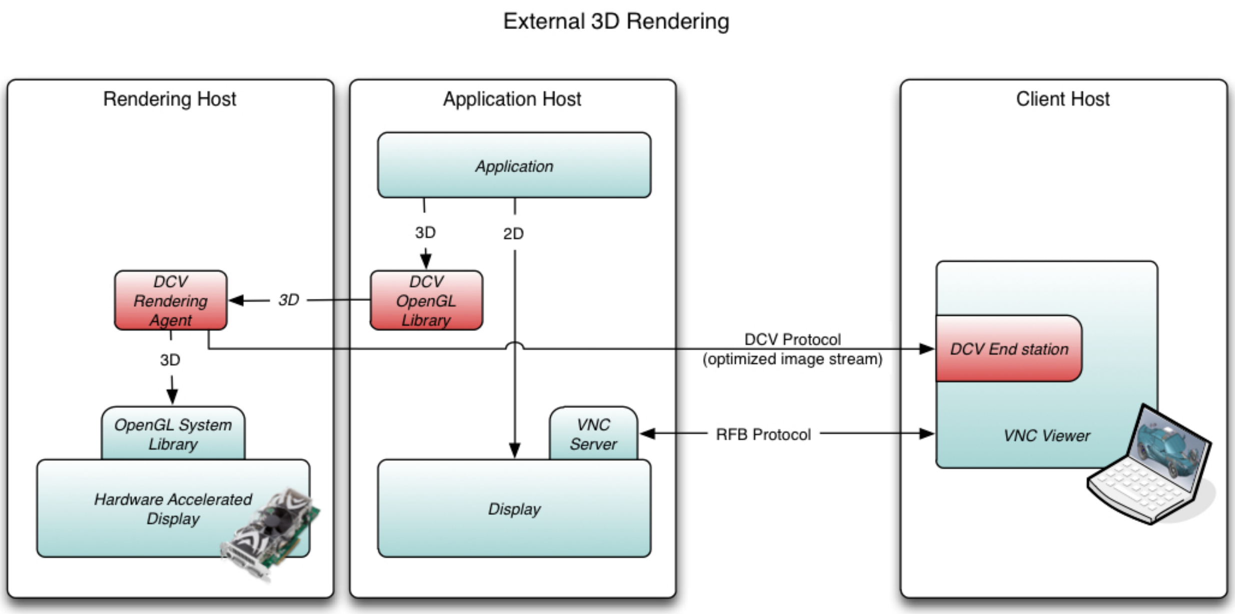


*Figure 1.2. NICE External Rendering Server*

Figure 1.2, "NICE External Rendering Server" illustrates DCV configured to delegate 3D rendering to a rendering host.

# 2 Prerequisites

DCV requires the following hardware systems:

- a physical application host machine equipped with 3D accelerated video adapters and capable of running the OpenGL or DirectX applications;

  or a virtual application host machine equipped with 3D accelerated video adapters using GPU pass-through and capable of running the OpenGL or DirectX applications;

- 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.

## Server Requirements

DCV is currently supported on the following server systems.

*Table 2.1. Server Requirements*

| **Server Requirements** | |
|---|---|
| Operating system *(Application Host)* | • Red Hat Enterprise Linux 5.x, 6.x, 7.0, 7.1 32/64-bit<br>• SUSE Enterprise Server 11 SP2 32/64-bit<br>• Oracle Linux 6.x 64-bit |
| | • Microsoft Windows 7 32/64-bit<br>• Microsoft Windows 8, 8.1 32/64-bit<br>• Microsoft Windows Server 2008 R2 32/64-bit<br>• Microsoft Windows Server 2012 R2 32/64-bit |
| | *With NICE External Rendering Server*<br>• Microsoft Windows 7 Professional 64-bit |
| Operating System *(Rendering Host)* | • Red Hat Enterprise Linux 6.x 64-bit |
| Graphics card and drivers | • NVIDIA Quadro family of *ultra high-end* and *high-end* graphics adapters<br>• NVIDIA GRID<br>• NVIDIA Tesla M2070-Q.<br>To download the latest NVIDIA graphic adapter drivers, go to the NVIDIA support site at http://www.nvidia.com. |

### Important

Some Linux desktop environments, like GNOME 3 (the default desktop environment used by Red Hat Enterprise Linux 7.x), do not work under DCV Server. In such case, we recommend installing and using an alternative desktop environment. For more information please refer to the section called “Linux desktop environment cannot be loaded”.

### Important

Different versions of RealVNC Server (such as VE 4.5.x or 5.x) or VNC software from other vendors are *not* compatible with this DCV Server release. The installer will ask to uninstall any incompatible version.

## Additional Requirements for NICE External Rendering Server

When used with External 3D Rendering configuration, DCV requires the following systems:

- a physical Linux rendering host equipped with 3D accelerated video adapters;
- a Windows application host machine (physical or virtual) capable of running the OpenGL applications (DirectX not supported);
- 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 and between rendering host and the end station.
- a *high bandwidth and low latency network* connection between the rendering host and the application host.

  The required bandwidth and maximum latency may vary from application to application and according to the size and complexity of the 3D data.

  Due to tight networking requirements, it is suggested to keep the application and rendering hosts on the same physical machine and use a virtual network link between them.

The configurations decribed in the following sections have been tested and are known to be working.

### NICE External Rendering Server with KVM

- physical Linux Rendering host running KVM hypervisor;
- virtual Windows Application hosts running as guests on the KVM hypervisor;
- Host-only Virtual Network between rendering host and application hosts using VirtIO network.

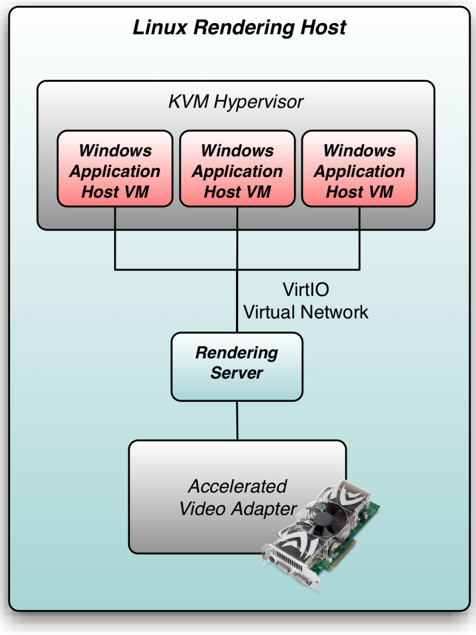

Linux Rendering Host
KVM Hypervisor
Windows Application Host VM
Windows Application Host VM
Windows Application Host VM
VirtIO Virtual Network
Rendering Server
Accelerated Video Adapter

*Figure 2.1. External 3D Rendering with KVM*

Figure 2.1, “External 3D Rendering with KVM” illustrates the typical deployment on KVM hypervisors.

## NICE External Rendering Server with XenServer

- virtual Linux Rendering host running on XenServer hypervisor with GPU pass-through;
- virtual Windows Application hosts running as guests on the same XenServer hypervisor;
- Host-only Virtual Network between rendering host and application hosts using optimized network drivers from XenServer Tools.

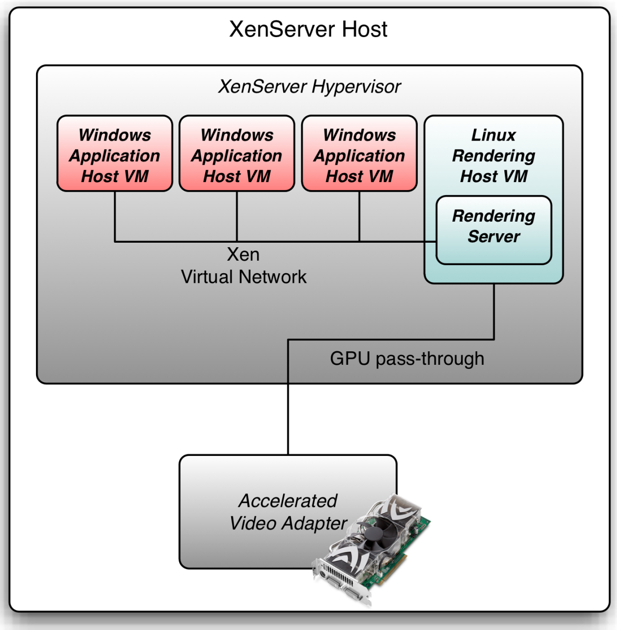


*Figure 2.2. External 3D Rendering with XenServer*

Figure 2.2, “External 3D Rendering with XenServer” illustrates the typical deployment on XenServer hypervisors.

## End Station Requirements

*Table 2.2. End Station Requirements*

| End Station Requirements | |
|---|---|
| Operating system | • Red Hat Enterprise Linux 5.x, 6.x, 7.x 32/64-bit<br>• SUSE Enterprise Server 11 SP2 32/64-bit |
| | • Microsoft Windows 7 32/64-bit<br>• Microsoft Windows 8, 8.1 32/64-bit<br>• Microsoft Windows 10 32/64-bit |
| | • Mac OS X Snow Leopard (10.6), Lion (10.7), Mountain Lion (10.8), Mavericks (10.9), Yosemite (10.10) |

The DCV installer includes and automatically installs RealVNC Visualization Edition 4.6.x (Viewer).

Portable end station packages which do not require installation are available. See the section called “Using Portable End Stations” for more information.

The application host and its end stations can run different operating systems.

Different versions of DCV Server and DCV End Station are compatible but some features may not be available when not using the latest version.

DCV is compatible with many *plain-VNC* clients from third parties. When using such clients, 3D images are delivered to the clients using the standard VNC protocol and therefore with reduced perfomance.

## Network Requirements

DCV is a network aware technology and may require extra firewall configuration. Your firewall should allow traffic on the interval of TCP ports between `7300` and `7399`. DCV uses ports in this interval to establish a connection for the 3D channel.

The ports can be changed in the `dcv.conf` configuration file (see Appendix A, *`dcv.conf`*).

RealVNC requires its own ports to open 2D channel. In a standard configuration your firewall should also allow traffic on:

- (Windows) TCP port `5900`
- (Linux) the interval of TCP ports between `5900` and `5999`

Please refer to RealVNC documentation for further information.

*Table 2.3. Summary of default firewall configuration*

| Host | Process | Rule |
|---|---|---|
| Application Host | RealVNC Server | (Linux) Allow incoming TCP connections on 5900-5999 ports from end stations<br><br>(Windows) Allow incoming TCP connections on 5900 port from end stations |
| | OpenGL processes (Linux) / `dcvd` process (Windows) | (Linux) Allow incoming TCP connections on 7300-7399 ports from end stations<br><br>(Windows) Allow incoming TCP connections on 7300 port from end stations |

**Note**

In addition to local firewall configuration check no external firewall is blocking the communication between end stations and application host.

### Additional Requirements for NICE External Rendering Server

When using DCV with NICE External Rendering Server configuration:

- the 3D channel uses connections from the end station to the *rendering host* (instead of the application host);
- the DCV rendering server uses TCP connections from the *application host* to the *rendering host* (port: `2007`);
- the OpenGL applications use TCP connections from the *application host* to the *rendering host* (ports: starting from `34567`, one port for each DCV Rendering Agent).

The network between the rendering server and application hosts does not need to be exposed to other hosts and therefore the suggested network topology is:

- `Host-Only Network`: an isolated network between rendering host and application hosts;
- `Public Network`: a network reachable to rendering hosts, application hosts and end stations.

*Table 2.4. Summary of default firewall configuration for External 3D Rendering*

| Host | Network | Process | Rule |
|---|---|---|---|
| Rendering Host | Public | DCV Rendering Agents | Allow incoming TCP connections on 7300-7399 ports from end stations |
| | Host-Only | DCV Rendering Server | Allow incoming TCP connections on 2007 port from application hosts<br><br>(See Note) |
| | Host-Only | DCV Rendering Agents | Allow incoming TCP connections on 34567 port and above (one port for each agent) from application hosts<br><br>(See Note) |
| Application Host | Public | RealVNC Server | (Windows) Allow incoming TCP connections on 5900 port from end stations |
| | Host-Only | OpenGL Applications | Allow outgoing TCP connections to rendering host on 2007 port<br><br>(See Note) |

**Note**

Since the Host-Only Network is an isolated network it is suggested to shut down on both application host and rendering host the firewall on the Host-Only network interfaces.

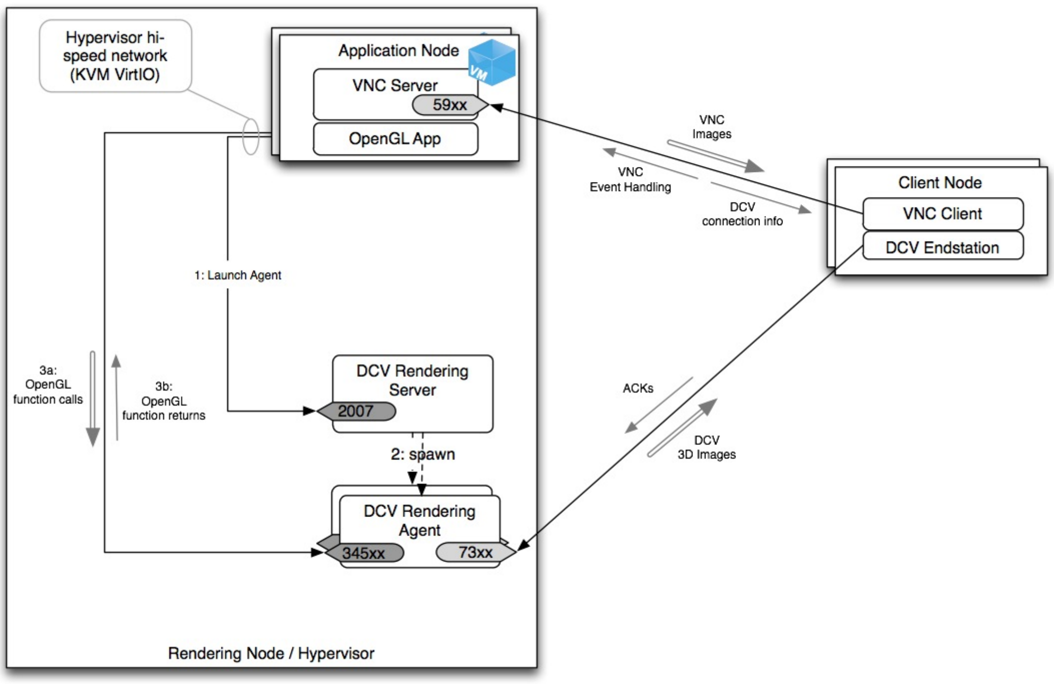


*Figure 2.3. DCV with External 3D Rendering*

Figure 2.3, "DCV with External 3D Rendering" illustrates the typical deployment when DCV works in External 3D Rendering configuration.

## X Server Requirements

On Linux, the host equipped with 3D accelerated video adapters must run an accelerated X Server configured according to these requirements:

1. NVIDIA drivers must be correctly working on this display.

   **Note**

   Official NVIDIA Drivers are required, the open source *nouveau* drivers are *not* supported. Please check the section called "NVIDIA Display Problems" for more information;

2. in case of multiple GPUs on the same node the suggested configuration is a single X Server for all GPUs, one Screen for each GPU.

   Example: `:0.0` for the first accelerated GPU, `:0.1` for the second accelerated GPU... Please check the section called "NVIDIA Display Problems" for more information.

   **Note**

   Alternative X setups are possible as long as DCV is able to open a connection with the X server managing the GPU card. For instance, when a system contains a heterogeneous set of GPUs, an administrator may prefer to configure a separate X server for each GPU instead of managing them as different screens of the same X Display.

3. local UNIX connections to the 3D accelerated X Server displays must be granted to the users of the 3D applications.

   DCV libraries run inside the 3D applications launched by the users. They redirect 3D calls to the accelerated display and so the applications processes run by the users must be able to access it.

   When running `dcvadmin enable`, DCV automatically searches for well-known X or display manager startup scripts and adds a call to `/opt/nice/dcv/bin/dcvxgrantaccess`. By default `dcvxgrantaccess` executes `xhost +local:`. So when X is launched the display access is granted.

   It is possible to change this behaviour, for example to restrict the access to a subset of users, providing a custom implementation of `dcvxgrantaccess`.

   It is important to understand the security risks implied by this configuration: users having access to a display could send or receive keystrokes and grab images from that display. Therefore it is important to never use the 3D accelerated X Server to login in, expecially as root.

   Please refer to the section called "Grant Local Connections to the 3D Rendering Display" for more information on this topic;

4. color depth must be 24-bit.

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

```
Section "Screen"
  Identifier     "Screen0"
  Device         "Device0"
  Monitor        "Monitor0"
  DefaultDepth    24
  SubSection     "Display"
    Depth       24
  EndSubSection
EndSection
```

5. only when using *Native Display mode* on Linux with VNC in *Service mode* additional configuration is required to enable the `vnc.so` module. Please refer to "Native X display support" documentation from RealVNC available here: http://www.realvnc.com/products/vnc/documentation/4.6/unix/x0.html#x0vncserver

## RealVNC Visualization Edition

DCV includes RealVNC Visualization Edition (RealVNC VE).

RealVNC VE is an enhanced version of RealVNC Enterprise Edition designed exclusively for DCV. The use of other VNC products (whether from RealVNC or another vendor) is not supported.

For detailed documentation about RealVNC VE refer to the documentation of the corresponding version of RealVNC Enterprise Edition.

For more information about RealVNC VE, go to http://www.realvnc.com/products/vnc/documentation/4.6.

3

# Installing DCV Server

## Before You Begin

Different releases of DCV cannot coexist on the same system. In case a previous version of NICE DCV is already installed you can upgrade it following the instructions below. In case of problems it is suggested to uninstall any earlier release, reboot the machine and then install the new release.

To remove earlier releases on Windows you can use the Windows Control Panel. To remove packages from your system on Linux:

- for DCV 2012 or earlier, you can use the **rpm -e nice-dcv-server** command
- starting from DCV 2013, you can use the uninstaller in **/opt/nice/dcv/bin/dcvuninstall**

The RealVNC Visualization Edition Server component required by DCV is embedded in the DCV Server installer. The DCV installer will detect RealVNC Visualization Edition Server and install it, in case it is not already available on the system.

Different versions of VNC on the same system may conflict with each other. It is suggested to uninstall any VNC component different from RealVNC Visualization Edition from the system before installing DCV.

See the following sections for OS-specific installation instructions.

## Installing on Linux

### Important

The DCV installer relies on the distribution package manager to install the required dependencies. Please double check your system has a working `yum` (on Red Hat) or `zypper` (on SuSE) installation and can access the default distribution packages. DCV requires only packages available in the official Linux distribution DVD. You must configure the Operating System installation DVD as the repository source in case you do not have access to the official Red Hat or SuSE repositories.

To install or upgrade a Linux application or rendering host perform the following steps:

1. log in as `root` on the node via SSH or directly from the console. Switch to runlevel 3:

```
# init 3
```

2. run the installer and follow the instructions:

```
# sh nice-dcv-2016.0-xxxx.run
```

The installer will perform checks including the NVIDIA card and driver presence. In case the NVIDIA driver is not found, the installer can download the latest version from NVIDIA website and install it automatically. If the host is not connected to the Internet or you want to provide your own version of the NVIDIA driver, place the NVIDIA driver installer in the same directory of the DCV one and launch the DCV installer.

The DCV installer provides a kernel module to remotize USB devices. If DKMS is installed on the system, the installer will ask whether to handle the module update using DKMS. If the administrator wants to have more control over the kernel module updates, the script `dcvusbdriverupdate` is provided. Call this script every time the kernel is updated.

At the end of the installation, the diagnostic script `dcvdiag` will be automatically launched. Check the output and, in case of any error or warning, apply the suggested fixes. A log of the installation, including the diagnostic information, will be saved and its location printed in the last page of the installer.

On successful completion of the installation, the files are installed in the `/opt/nice/dcv` directory.

### Note

DCV is automatically enabled on completion of the installation process. In case you update your kernel version or graphics drivers, run the **dcvadmin enable** command as `root` to re-enable the software.

Refer to the section called “NVIDIA Display Problems” for a list of cases in which DCV needs to be enabled again.

## Installing on Windows

**Note**

Upgrading from previous releases is not supported by the installer. Please uninstall the old packages before installing the new ones.

To install an application host perform the following steps:

1. log in as an administrator user via Remote Desktop Connection or directly from the console;
2. run the appropriate installer depending on your system architecture and follow the displayed instructions:
   - on 64-bit systems: `nice-dcv-server-2016.0-xxx-x64.msi` (includes 32-bit binaries too).
   - on 32-bit systems: `nice-dcv-server-2016.0-xxx-x86.msi`

On successful installation, if you selected the default installation path, the software will be located at `<Program Files>\NICE\DCV`.

The installer modifies the system PATH environment variable.
To generate logs of the installation process, open a Command Prompt as Administrator and launch the installer in the following way:

```
msiexec /i <path-to-msi> /log C:\install.log
```

**Note**

DCV is automatically enabled on completion of the installation process.

On Windows with External Rendering Server, before installing service packs or upgrading the NVIDIA drivers, temporarily disable DCV functionality:

1. run **dcvadmin disable** at the command prompt;
2. complete the service pack or driver installation;
3. re-enable DCV using the **dcvadmin enable** command.

Refer to the section called “Verifying Installation” to verify your installation is working properly.

## Installing License

DCV supports two different license configurations:

- `demo:` a demo license file must be installed on each application host;
- `floating:` a license server distributes licenses to application hosts. A license file must be installed on the license host. Application hosts must be configured to check-out licenses from the license server.

NICE DCV licensing module is based on Reprise License Manager (`http://www.reprisesoftware.com`).

You do not need to install a license key on DCV End Stations.

You do not need to install a license key on DCV Rendering Hosts.

The NICE DCV installer will ask you to provide the path to the demo license file or the network address of the license server.

If you already installed NICE DCV on your application host, to install a demo license, you simply need to rename the license file you received from NICE to license.lic and copy it to the following directory:

1. `/opt/nice/dcv/license/` on Linux application hosts.
2. `<Program Files>\NICE\dcv\license\` on Windows application hosts;

To replace an old demo license with a new one, you can simply replace the old license.lic file with a new one.

### Note

The specified path could be different in case you installed NICE DCV in a different location. The license file must be copied to `DCV_ROOT\license` where DCV_ROOT is the NICE DCV installation directory. Note that administrative rights are needed to access the `<Program Files>\NICE\dcv\license\license.lic` file.

For production installations, configure the application hosts to connect to the RepriseLM server:

1. open the `DCV_ROOT/conf/dcv.conf` file with a text editor;
2. add to the `[License]` section the following line and change to point it to your RepriseLM server:

```
license_file = 5053@myRLMserverhost
```

You can request a license to your NICE commercial contact or to <`helpdesk@nice-software.com`>.

If you already have an instance of Reprise License Manager deployed on your infrastructure, you can simply add the `nice.set` file to it, otherwise you will need to deploy the license server.

Please refer to the Reprise License Manager (`http://www.reprisesoftware.com`) documentation for more information about installing, deploying and managing the license server on your infrastructure.

For convenience, the DCV ISO Image contains a copy or the Reprise License Manager server software.

**Note**

Detailed instructions on how to set up Reprise License Manager are out of the scope of this document and are better left to the original Reprise License Manager documentation. However there are some important considerations and good practices that are worth pointing out:

- The license server host needs to be reachable by the DCV application hosts, but not by the end stations. It is a good practice to not expose the license server on the Internet and to make sure it is properly protected by a firewall. <listitem></listitem>

  Never run the license server as root, always use an unpriviledged account. <listitem></listitem>

  The Reprise License Manager server includes an embedded Web Server to provide an user friendly interface. For maximum security, run the license server with the `-nows` command line option that disables the Web Server. If you prefer to keep the Web Server, make sure it is properly secured by configuring its authentication mechanisms. Refer to the Reprise License Manager documentation for more information.

# Advanced Installations

This section describes advanced installation topics.

## Windows Unattended Installation

To deploy the NICE DCV in a unattended mode you can use the Windows tool `msiexec`, with the following switches:

```
C:\> msiexec /QN /I nice-dcv-server-2016.0-xxx-xx.msi
```

To remove the NICE DCV use:

```
C:\> msiexec /QN /X nice-dcv-server-2016.0-xxx-xx.msi
```

or the WMI command-line utility, executing:

```
C:\> wmic product where name="NICE Desktop Cloud Visualization" call uninstall
    /nointeractive
```

At the end of the both installation and uninstallation processes, the machine will reboot.

By default the installer configures DCV to search for a demo license in `DCV_ROOT\license\license.lic`. If you want to use a demo license please copy it in this location.

Alternatively you can use the `LICENSE_SERVER`, `LICENSE_SERVER_HOST` and `LICENSE_SERVER_PORT` parameters to instruct the installer to configure DCV to use a specific license server as in the following example:

```
C:\> msiexec /QN /I nice-dcv-server-2016.0-xxx-xx.msi LICENSE_SERVER=1
            LICENSE_SERVER_HOST=10.10.10.10 LICENSE_SERVER_PORT=5053
```

To install NICE DCV with External Rendering Server you *must* specify `EXT_RENDERING_SERVER=1` and set `DCV_RENDERING_SERVER` to the network address of the rendering server:

```
C:\> msiexec /QN /I nice-dcv-server-2016.0-xxx-xx.msi
            EXT_RENDERING_SERVER=1 DCV_RENDERING_SERVER=10.10.10.11
```

## Installing on Linux from RPMs

The installer takes care of installing all required packages and dependencies and is the suggested way to install DCV. However, if you prefer to install each component using your preferred package manager you can find the RPMs inside the DCV ISO Image which can be downloaded NICE web site.

To install or upgrade an application host, log in as `root` and perform the following steps:

1. from a terminal go to the directory containing the installation files;
2. install the appropriate packages depending on your distribution:
    - RHEL 32 bit:

```
# yum install --nogpgcheck vnc-VE4_6_3-x86_linux.rpm \
                     nice-dcv-server-2016.0-xxx.i686.rpm
```

- RHEL 64 bit:

```
# yum install --nogpgcheck vnc-VE4_6_3-x64_linux.rpm \
                     nice-dcv-server-2016.0-xxx.i686.rpm \
                     nice-dcv-server-2016.0-xxx.x86_64.rpm
```

- SLES 32 bit:

```
# zypper install -f vnc-VE4_6_3-x86_linux.rpm \
                     nice-dcv-server-2016.0-xxx.i686.rpm
```

- SLES 64 bit:

```
# zypper install -f vnc-VE4_6_3-x64_linux.rpm \
                     nice-dcv-server-32bit-2016.0-xxx.i686.rpm \
                     nice-dcv-server-2016.0-xxx.x86_64.rpm
```

### Note

DCV is automatically enabled on completion of the installation process. In case you update your graphics drivers, run the **dcvadmin enable** command as `root` to re-enable the software. Refer to the section called “NVIDIA Display Problems” for a list of cases in which DCV needs to be enabled again.

On successful completion of the installation, the files are installed in the `/opt/nice/dcv` directory.

In case of upgrade the old `dcv.conf` configuration file will be preserved. The configuration file included in the installer is installed for reference as `dcv.conf.rpmnew`.

Refer to the section called “Verifying Installation” to verify your installation is working properly.

## Configure a Rendering Hosts

To configure an existing Linux DCV installation to work as an External Rendering Server follow these steps:

1. open the `DCV_ROOT/conf/dcv.conf` file;
2. in the `[RenderingServer]` section specify the `host` value to the IP address of the low latency/high bandwidth network interface to which the rendering server should bind. If not specified, the rendering server process will bind to all available interfaces
3. in the `[RenderingServer]` section set the `local_display_list` value with a space separated list of local X accelerated displays to use. If not specified, the default value is `:0.0`

4. in the `[Remotization]` section set the `host` value to the IP address on the network interface reachable from the end station. If not specified the hostname of the rendering host will be used

5. as the `root` user, start the DCV rendering server process on the node:

   ```
   # /opt/nice/dcv/etc/init.d/dcvrenderingserver start
   ```

   The `dcvrenderingserver` process will run as the `dcv` unpriviledged user.

It is suggested to install the dcvrenderingserver as a system service. As the `root` user run the following steps:

1. link the start the dcvrenderingserver script from DCV /etc/init.d:

   ```
   # ln -s /opt/nice/dcv/etc/init.d/dcvrenderingserver
                /etc/init.d/dcvrenderingserver
   ```

2. add the dcvrenderingserver to system services:

   ```
   # chkconfig --add dcvrenderingserver
   ```

4

# Installing DCV End Station

The installation of the DCV End Station requires administrative access to the client host. In case administrative priviledges are not available, portable packages which do not require any installation can be used. See the section called “Using Portable End Stations” for more information.

See the following sections for OS-specific installation instructions.

## Installing Windows End Stations

### Important

DCV End Station embeds a component called RealVNC Visualization Edition Viewer.

Different versions of VNC on the same system may conflict with each other. It is suggested to uninstall any VNC component from the system before installing DCV.

To install the DCV End Station perform the following steps:

1. download the DCV End Station MSI archive for your system architecture from `https://www.nice-software.com/dcv`
2. navigate to the folder containing the installation files
3. run the installer as Administrator and follow the displayed instructions
4. log off and log in again. This action updates the system configuration so that you can start using the software.

### Note

To generate logs of the installation process, open a Command Prompt as Administrator and launch the installer in the following way:

```
msiexec /i <path-to-msi> /log C:\install.log
```

## Installing Mac End Stations

To install the DCV End Station, perform the following steps:

1. download from `https://www.nice-software.com/dcv` the DCV End Station DMG archive
2. open the installer DMG
3. drag and drop the DCV End Station application from the DMG to the Applications folder
4. to always open `.vnc/.dcv` files using the DCV End Station press `CMD+I` on a `.vnc/.dcv` file, select `Open with` DCV End Station and press `Change all`.

To access the `dcv.conf` configuration file right click on the DCV End Station application, select `Show Package Contents` and navigate to the `Contents/Resources/conf` directory.

### Note

In case of upgrade the full application contents, including the `dcv.conf` file, will be replaced with the defaults. Remember to backup the configuration in case you changed it.

## Installing Linux End Stations

To install the DCV End Station, perform the following steps:

1. download the DCV End Station *.run* archive from `https://www.nice-software.com/dcv`
2. from a terminal go to the directory containing the installation files
3. run the installer as the `root` user and follow the instructions:

```
# sh nice-dcv-endstation-2016.0-xxx.run
```

# Using Portable End Stations

DCV Portable End Station is a bundled version of DCV End Station including everything is needed to remotely connect to a DCV Server. It does not require an installation making it easy to use from any client computer.

**Note**

USB remotization requires to install drivers on the system and will not be available when using a portable end station.

## Using Windows Portable End Station

To use the DCV Portable End Station, perform the following steps:

1. download the DCV Portable End Station ZIP archive from `https://www.nice-software.com/dcv`
2. expand the archive to a local folder or a USB stick and open its contents
3. double click on `niceviewer` to start the DCV End Station.
4. optionally, double click on `setup-niceviewer.bat` to always open `.vnc/.dcv` files with this DCV Portable End Station

## Using Linux Portable End Station

To use the DCV Portable End Station, perform the following steps:

1. download the DCV Portable End Station `tar.gz` archive from `https://www.nice-software.com/dcv`
2. expand the archive to a local folder with:

```
$ tar xvzf nice-dcv-portable-2016.0-xxx.tar.gz
```

3. to launch the DCV End Station run:

```
$ niceviewer
```

## Using Mac Portable End Station

The DCV End Station provided in the DMG package can be used directly without installing it.

# Advanced Installations

## Windows Unattended Installation

To deploy the NICE DCV End Station in a unattended mode you can use the Windows tool `msiexec`, with the following switches:

```
C:\> msiexec /QN /I nice-dcv-endstation-2016.0-xxx-Release.msi
```

To remove the DCV End Station you can use the Windows tool `msiexec`, with the following switches:

```
C:\> msiexec /QN /X nice-dcv-endstation-2016.0-xxx-Release.msi
```

or the WMI command-line utility, executing:

```
C:\> wmic product where name="NICE Desktop Cloud Visualization Endstation"
      call uninstall /nointeractive
```

At the end of the both installation and uninstallation processes, the machine will reboot.

## Installing Linux End Stations from RPMs

To install a DCV End Station, perform the following steps:

1. download the DCV End Station RPM archive for your system architecture from `https://www.nice-software.com/dcv`
2. from a terminal go to the directory containing the installation files
3. as the `root` user, install the appropriate packages depending on your distribution:

   - RHEL 32 bit:

     ```
     # yum install --nogpgcheck vnc-VE4_6_3-x86_linux.rpm \
                         nice-dcv-endstation-2016.0-xxx.i686.rpm
     ```

   - RHEL 64 bit:

     ```
     # yum install --nogpgcheck vnc-VE4_6_3-x64_linux.rpm \
                         nice-dcv-endstation-2016.0-xxx.x86_64.rpm
     ```

   - SLES 32 bit:

     ```
     # zypper install -f vnc-VE4_6_3-x86_linux.rpm \
                         nice-dcv-endstation-2016.0-xxx.i686.rpm
     ```

   - SLES 64 bit:

```
# zypper install -f vnc-VE4_6_3-x64_linux.rpm \
                    nice-dcv-endstation-2016.0-xxx.x86_64.rpm
```

## Configuring Proxy Support

DCV End Station supports connection to a server behind an HTTP or SOCKS5 proxy.

To enable proxy support you need to specify the proxy type, server address and port in the `dcv.conf` file. Please refer to Appendix A, *`dcv.conf`* for more information.

In the following `dcv.conf` excerpt, the end station is configured to connect to an `HTTP` proxy running on port `3128` (the default port for Squid server) on node `proxy.intranet`.

```
[Endstation]
proxy_port=3128
proxy_server=proxy.intranet
proxy_type=HTTP
```

In the following example, the end station connects to a `SOCKS5` proxy running on port `1080` (the default port for Antinat server) on node `proxy.intranet`.

```
[Endstation]
proxy_port=1080
proxy_server=proxy.intranet
proxy_type=SOCKS5
```

To enable the proxy support in RealVNC please refer to the RealVNC documentation available here: http://www.realvnc.com/products/vnc/documentation/4.6/docs/Chapter3.html in the "Configuring VNC Viewer before you connect" paragraph.

## Configuring Encryption Support

DCV supports encryption of the 3D channel between the end station and the server.

By default, an end station uses the server encryption preferences. This chapter explains how the end station can be configured to tighten the encryption rules if supported by the server.

You can configure DCV End Station using the `dcv.conf`.

The `encryption` setting in the `[Endstation]` section supports the following values:

- `Server` *(default)*: uses the server's encryption preferences.
- `PreferOff`: use an unencrypted connection if possible.
- `PreferOn`: use an encrypted connection if possible.
- `AlwaysOn`: do not connect to servers that do not use at least 128-bit encrypted connections.
- `AlwaysMaximum`: do not connect to servers that do not use 256-bits encrypted connections.

By default RealVNC communication between client and host computer is encrypted using industry-standard 128-bit AES. For more information on how to request a different encryption option, see

Chapter 2 "Getting Connected", "Step 4: Request an encryption option", http://www.realvnc.com/products/vnc/documentation/4.6/docs/ae1014953.html.

For more information on RealVNC security, see Chapter 7 "Securing Connections", http://www.realvnc.com/products/vnc/documentation/4.6/docs/Chapter7.html.

# 5 Advanced Topics

## USB Remotization

NICE DCV USB Remotization allows users to plug USB devices on their client machine and use them on the remote desktop.

Limitations:

- Works only for DCV end stations installed on Windows 7, 8 or 8.1.
- Portable end stations are not supported
- Tested only with 3D mice and memory sticks

When the end station connects to a server with the USB Remotization feature enabled, the DCV Console shows the option to share local USB devices.

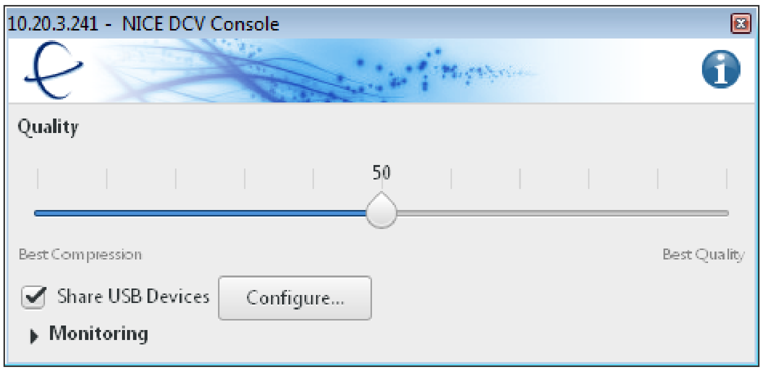


*Figure 5.1. NICE DCV Console with USB Remotization*

For security reasons, DCV provides a white list of USB device classes which are allowed to be remotely connected. By default this list contains USB mass storage devices and well known 3D mice.

Devices can be added or removed by editing the `DCV_ROOT\share\dcv\usb-devices.txt` file on the DCV Server.

The information to add to the white list can be retrieved using the `DCV_ROOT\bin64\dcvlistusb.exe` application on the client side. Attach the USB device to the client computer and launch the application. It will output the list of the currently attached USB devices:

```
List of devices (Name, Base Class, Subclass, Protocol, Vendor, Product):
Spaceball 5000, 3, -1, -1, 1133, 50721
Space Traveller 3D Mouse, 3, -1, -1, 1133, 50723
```

To allow a specific device, copy the line into the `usb-devices.txt` file:

```
Space Traveller 3D Mouse, 3, -1, -1, 1133, 50723
```

To allow a range of devices, place a `-1` on the fields to be ignored:

```
Mass storage, 8, -1, -1, -1 -1
```

Or add a line like this to allow any device:

```
Any Device, -1, -1 -1, -1, -1
```

## Smart Card Remotization

Starting from NICE DCV 2014.0, applications running in a remote Linux desktop can seamlessly access the local smart card, using the standard PC/SC interface. Users can use smart cards for encrypting their emails, signing documents and authenticating against remote systems.

The Smart Card Remotization feature is available only on Windows or Linux end stations connected to Linux servers.

When the end station connects, the DCV Console shows the option to share the local Smart Cards. See Figure 5.2, “NICE DCV Console”.

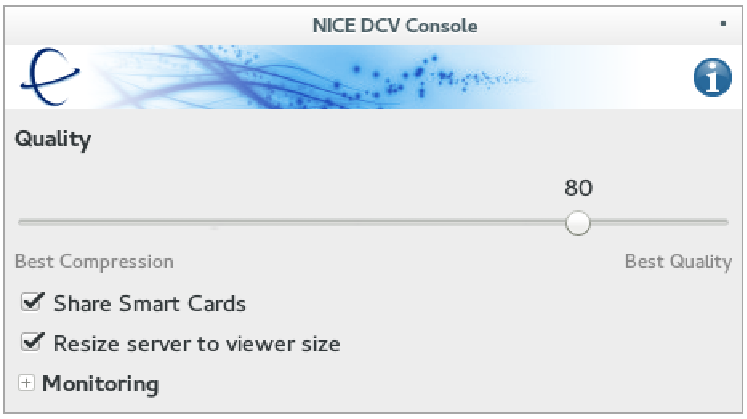
The `dcvscrun` (see the section called “Command Line Tools”) must be used to launch Linux applications which use the standard PC/SC interface so that they can use the DCV Smart Card Remotization.

The `dcvscrun` adds to the `LD_LIBRARY_PATH` the `DCV_ROOT/lib[64]/pcsc` libraries and then launches the desired application.

It is not required to launch applications via `dcvscrun` as long as the appropriate `LD_LIBRARY_PATH` is set in the application environment via:

```
DCV_ROOT=/opt/nice/dcv
export LD_LIBRARY_PATH="${DCV_ROOT}/lib64/pcsc:${DCV_ROOT}/lib/pcsc"
```

In some cases the setting of `LD_LIBRARY_PATH` can be added to the xstartup script so that all applications launched in the session will use the right environment. Note however one of the wrappers during a session startup may reset this setting. In particular any wrapper using the `setuid` function will reset the `LD_LIBRARY_PATH`.



☑ Share Smart Cards
☑ Resize server to viewer size

*Figure 5.2. NICE DCV Console*

## Adaptive Server-Side Resolution

Starting from NICE DCV 2014.0, the end station can automatically adapt the server-side screen resolution to the size of the viewer window.

The Adaptive Server-Side Resolution feature is available only on Windows or Linux end stations connected to Linux servers.

When the end station connects, the DCV Console shows the option to enable server size resizing. See Figure 5.2, “NICE DCV Console”.

In case the session is shared by multiple concurrent users this feature is disabled.

The resolution can vary in the range from 600x400 to 5120x2160. Trying to resize the client to a width or height outside of the limits has no effect.

This feature uses Xrandr support to resize the desktop. The available resolution range is specified at vncserver startup time with the `-randr` option.

You can change the default range by editing the `/etc/vnc/config` file or `${HOME}/.vnc/config`. Note the maximum size specified has an impact on the memory allocated by the Xvnc process.

## Audio Remotization

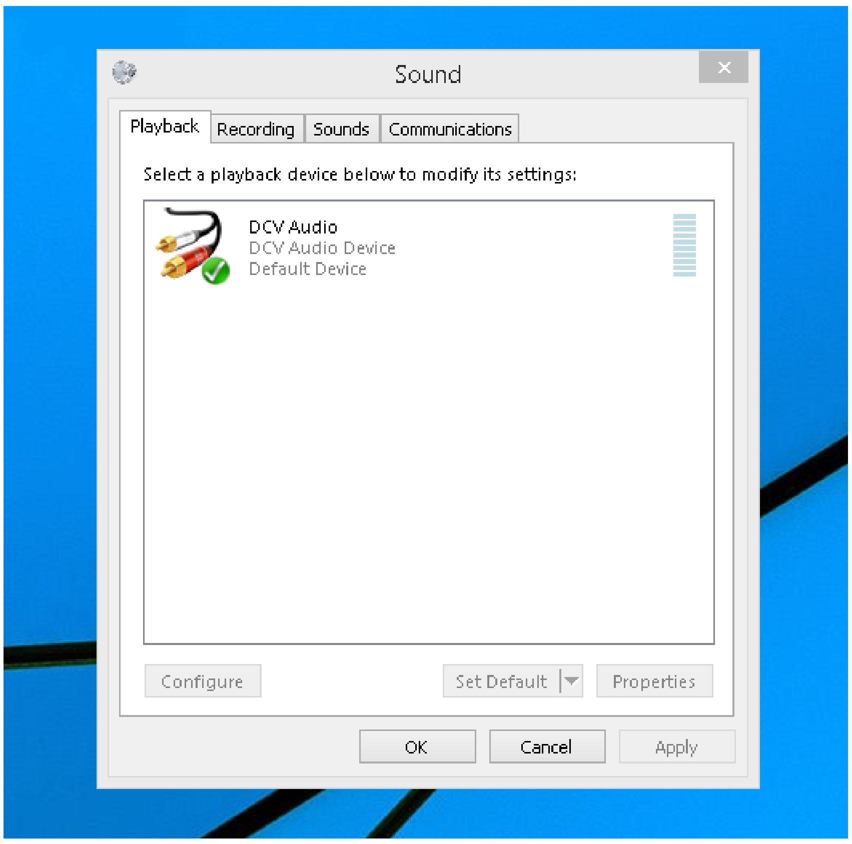
Starting from NICE DCV 2016.0, the end station can play back sounds and music generated by applications running on the remote desktop.

Limitations:

- Works only for DCV server and end stations installed on Microsoft Windows
- Only audio output is supported, sounds recorded by a microphone on the client machine currently are not sent to the server

When DCV server is running on a Windows session, it automatically adds and activates a virtual audio output device to the Windows system. Such device takes care of compressing and sending the audio stream to the end station, which then receives it and reproduces it on the local speakers.

The DCV audio device appears among the system playback devices.



Select a playback device below to modify its settings:

DCV Audio
DCV Audio Device
Default Device

*Figure 5.3. DCV audio device*

The DCV audio device can be controlled by the system mixer.

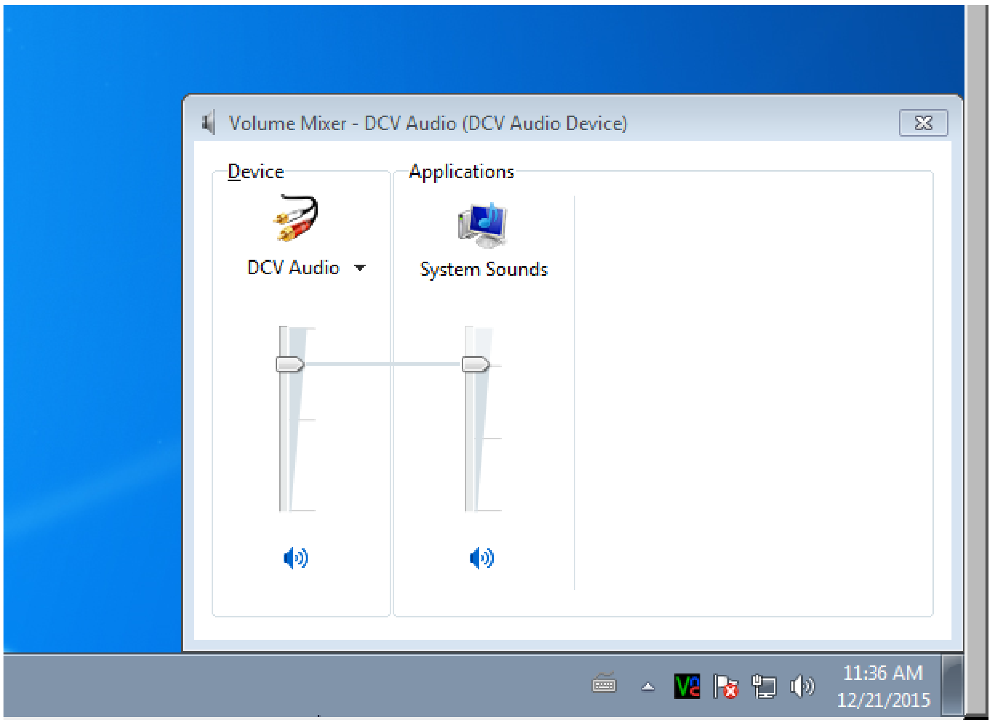


*Figure 5.4. DCV audio mixer*

## Proxy Support

DCV End Station can connect to a server behind a proxy supporting HTTP Connect or SOCKS5 protocols.

The proxy support can be configured directly on end stations or can be configured on the server which will push the configuration to the end stations.

Proxy configuration on end stations overrides the proxy configuration from the server.

To configure the server to push proxy configurations to end stations you need to specify the proxy type, server address and port in the `dcv.conf` file in the `[Remotization] section`. Please refer to Appendix A, *`dcv.conf`* for more information.

In the following `dcv.conf` excerpt, the server is configured to push as a default to end stations an `HTTP` proxy running on port `3128` (the default port for Squid server) on node `proxy.intranet`.

```
[Remotization]
default_proxy_port=3128
default_proxy_server=proxy.intranet
default_proxy_type=HTTP
```

In the following example, the server pushes a `SOCKS5` proxy running on port `1080` (the default port for Antinat server) on node `proxy.intranet`.

```
[Remotization]
default_proxy_port=1080
default_proxy_server=proxy.intranet
default_proxy_type=SOCKS5
```

### Important

The proxy push feature is not supported by older pre-2012.1 end stations and may prevent them to connect.

In case you want to use the proxy push feature but need to retain compatibility with older end stations (pre 2012.1) you must specify in the `[Remotization]` section the `host` parameter.

Additional configuration on version-2012.0 end stations is required to connect via proxy.

Version 2011.0 end stations do not support proxy and need direct access to the DCV Server to work.

To enable the proxy support in RealVNC please refer to the RealVNC documentation available here: http://www.realvnc.com/products/vnc/documentation/4.6/docs/Chapter3.html in the "Configuring VNC Viewer before you connect" paragraph.

# Encryption Support

DCV supports encryption of the 3D channel between the end station and the server.

By default, communication between end station and server is not encrypted. This chapter explains how to tighten the encryption rules if necessary.

You can keep this configuration if you are sure all potential client computers are within a secure network environment and all potential users are trustworthy. This also provides backward compatibility with older end stations and may improve performance in particular on low end client machines.

You can tighten the encryption rules by forcing AES with 128-bit keys or make connections ultra-secure forcing 256-bit keys. This may impact performance. It also means only new end stations can connect to the server.

The encryption actually used on the channel is the result of the handshake between the end station and the server.

### Note

A connecting end station can request that the encryption rules be tightened, but not relaxed.

You can configure DCV Server using the `dcv.conf` on the DCV Application Server (or the DCV Rendering Server in case you are using an External Rendering Server).

The `encryption` setting in the `[Remotization]` section supports the following values:

- `PreferOff` *(default)*: encryption is preferably off. Choose this option to allow older versions of DCV End Station to connect. A connecting user can request the encryption to be turned on, either to 128-bit AES (by setting encryption on end station to PreferOn or AlwaysOn), or 256-bit AES (by setting encryption on the end station to AlwaysMaximum).
- `PreferOn`: encryption is preferably on using 128-bit keys. A connecting user can request either to turn it off (by setting the encryption on the end station to PreferOff), or the AES key size be increased to 256-bit (by setting encryption on end station to AlwaysMaximum).
- `AlwaysOn`: encryption is always on. The AES key size is 128-bit or 256-bit if the connecting user requires a 256-bit encrypted connection.
- `AlwaysMaximum`: encryption is always on. The AES key size is always 256-bit. A connecting user cannot request the encryption to be turned off, or the AES key size reduced to 128-bit.

### Important

To ensure all information is encrypted:

- RealVNC server and viewer must be configured to use encryption;
- DCV Server must be configured to use AlwaysOn or AlwaysMaximum encryption.

Note this configuration will prevent older end stations to connect. To preserve backward compatibility with older end stations the server must be configured to PreferOn or PreferOff.

For more information on RealVNC security, see Chapter 7 "Securing Connections", http://www.realvnc.com/products/vnc/documentation/4.6/docs/Chapter7.html.

Please refer to the section called “Configuring Encryption Support” for information on encryption configuration on the end station.

## Remote Desktop Connection Support

Since version 2013.0, DCV supports OpenGL applications inside Microsoft Remote Desktop Connections.

Thanks to DCV virtual GPU technology it is now possible to run OpenGL applications in a Remote Desktop Session with full OpenGL acceleration, delegating the rendering to a Linux External Rendering Server.

Users can connect to the remote sessions using the widely available Microsoft Remote Desktop Client without the need to install any other component on the client.

If available, DCV can leverage the RemoteFX codec for the efficient delivery of 3D images.

### Prerequisites

Using DCV with Remote Desktop Connection requires:

- at least one NICE External Rendering Server, see the section called “Server Requirements” for more information
- at least one Windows Application Hosts configured to use the NICE External Rendering Server for OpenGL rendering, see the section called “Server Requirements” for more information

The same networking requirements between the Application Hosts and Rendering Hosts described in the section called “Additional Requirements for NICE External Rendering Server” apply here too.

For optimal performance Microsoft Remote Desktop Protocol 8.0 is required, which enables RemoteFX for the efficient delivery of 3D images. For more information on how to enable RDP 8.0 please refer to Microsoft documentation. The following Microsoft Knowledge Base article describes how to enable RDP 8.0 on Windows 7 SP1: `http://support.microsoft.com/kb/2592687`.

6

# Using Desktop Cloud Visualization

This chapter describes how to run remote OpenGL applications with DCV and use the DCV components both on application hosts and end stations, both on Linux and Windows.

## Using DCV on an End Station

### Using DCV on a Windows End Station

To start the end station on Windows, click `Start > All Programs > NICE > DCV > DCV Viewer`.

### Using DCV on a Mac End Station

Go to the Applications folder and launch the `DCV End Station`.

### Using DCV on a Linux End Station

To start the end station on Linux, click on the programs menu and select `Internet > VNC Viewer`.

To launch the end station from command line, use the `vncviewer` command.

This is the syntax of the `vncviewer` command is:

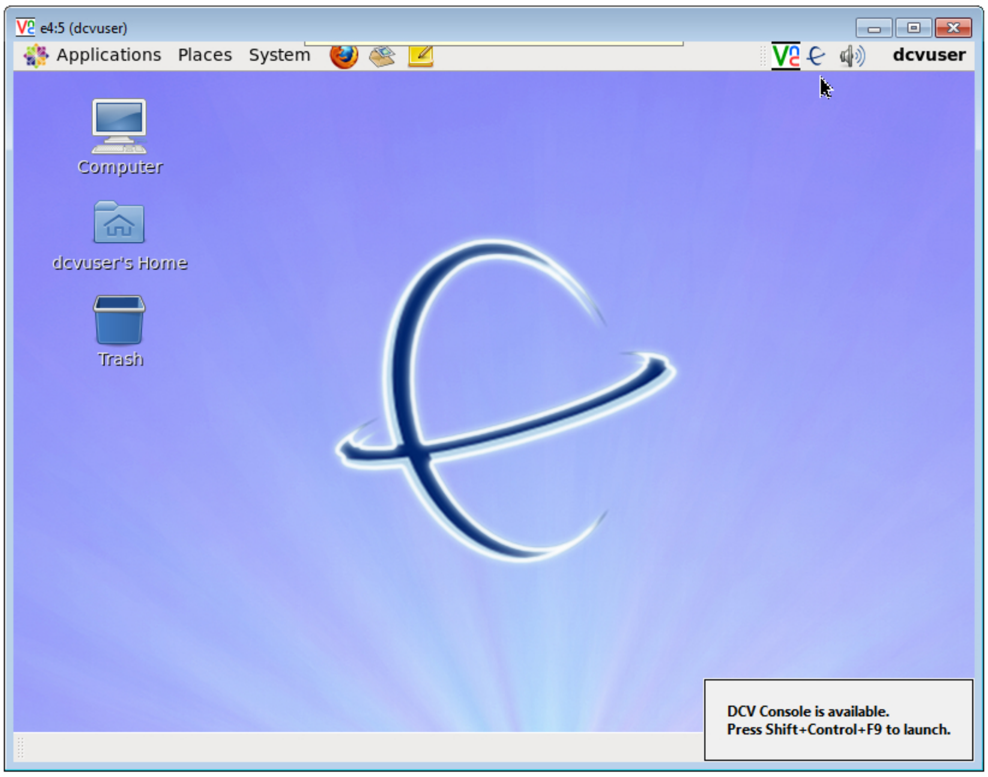
```
vncviewer [<servername[:portnum]>] [<other viewer options>] [-h]
```

Refer to the RealVNC documentation for more information about the `vncviewer` command.

**Important**

Make sure that the `vncviewer` program used is the one provided by RealVNC Visualization Edition, the DCV component will not be loaded by other VNC clients.

After connection to a remote server a tool-tip will pop-up showing the availability of the DCV Console as in Figure 6.1, “Example of a Linux Desktop at connection”.



DCV Console is available.
Press Shift+Control+F9 to launch.

*Figure 6.1. Example of a Linux Desktop at connection*

## Using DCV from a plain VNC client

On some clients it may be not possible to install the DCV End Station but a plain RealVNC Viewer may be available.

Starting from DCV 2012.2, the DCV Server automatically detects plain VNC clients (without DCV End Station) and sends the 3D images on their VNC connection. The DCV Config tool reports the number of currently connected plain VNC clients.

This feature is useful to enable access from iOS devices such as iPhones or iPads using the standard RealVNC client available from the Apple App Store.

### Important

When connecting from a DCV End Station double check the DCV Console tool-tip shows in the bottom right corner. In case it does not show, the DCV End Station may not be working properly and your connection may be not optimized.

## Using DCV Console

The DCV Console provides a graphical interface to control and monitor the DCV connection and to tweak its settings. The same interface is available both on Windows and Linux end stations.

Figure 6.2, “DCV Console main window” shows the main window of the DCV Console.

The DCV Console can be started by pressing the default `<Control><Shift>F9` keystroke combination in the VNC Viewer window. The keystroke combination can be configured in the `dcv.conf` file by setting the `console_shortcut` property. Please refer to Appendix A, *`dcv.conf`* for more information.

**Note**
The keystroke combination is `<CMD><Fn>F9` it is not possible to configure the keystroke combination on Mac.

Normally a user just needs to operate on the quality slider to adapt DCV to his current needs.

An in depth description of the available settings follows.

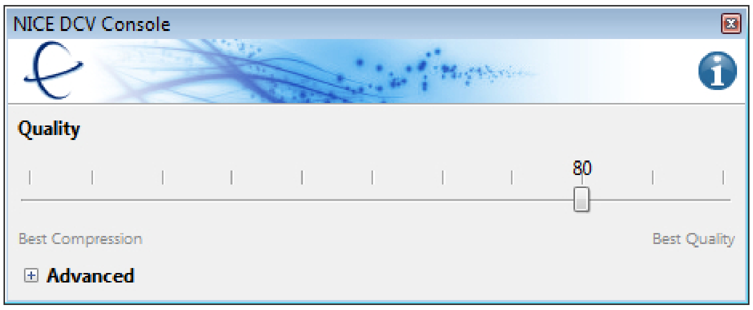


*Figure 6.2. DCV Console main window*

## Quality slider

Controls the balance between image compression and image quality. Choosing a high value improves the quality of the 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.

By changing the Quality slider you may notice a change in the bandwidth usage and/or FPS. Adjust it to find the value which works best in your environment. The best value may change depending on the application, the image complexity and the network conditions.

Starting from DCV 2013, you can easily switch to lossless compression by setting the quality slider to Best Quality as shown in Figure 6.3, “DCV Console main window showing lossless compression”. Due to the higher bandwidth usage of lossless compression, this setting is suggested only in LAN environments.

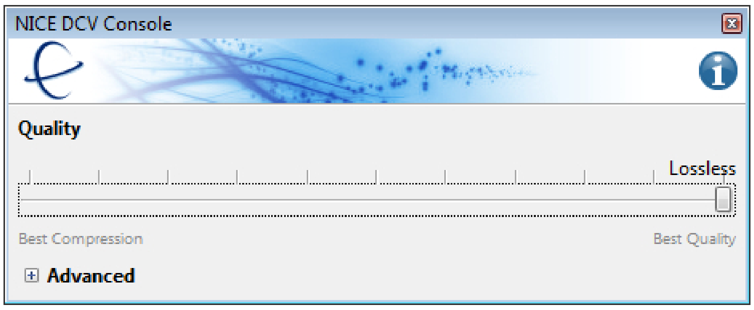


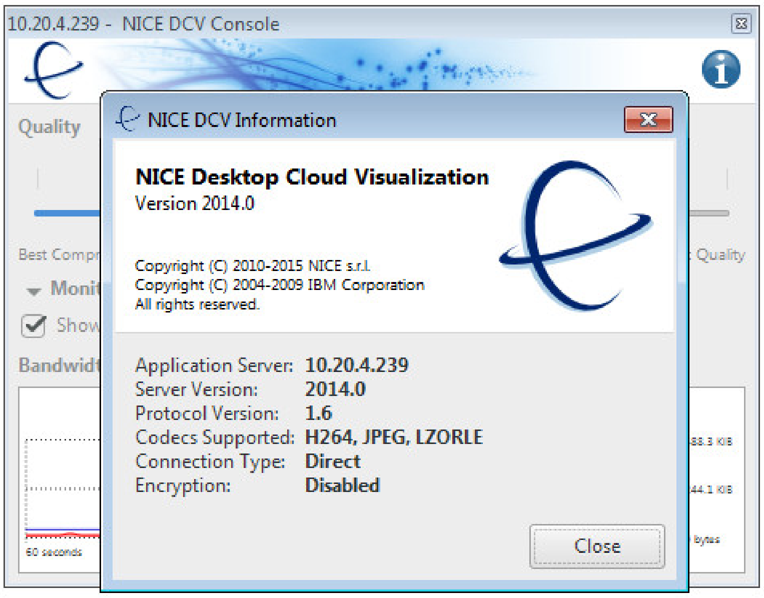
*Figure 6.3. DCV Console main window showing lossless compression*

## About

Clicking on the information icon on the top right shows the About dialog which contains information on the DCV End Station version, the DCV Server version and hostname, the DCV protocol, the codecs in use, the connection type (Direct or via a Proxy) and the used encryption.

These information are available after the first OpenGL is launched.

Figure 6.4, “DCV Console About window” shows the About window of the DCV Console.



NICE Desktop Cloud Visualization
Version 2014.0

Copyright (C) 2010-2015 NICE s.r.l.
Copyright (C) 2004-2009 IBM Corporation
All rights reserved.

| | |
|---|---|
| Application Server: | 10.20.4.239 |
| Server Version: | 2014.0 |
| Protocol Version: | 1.6 |
| Codecs Supported: | H264, JPEG, LZORLE |
| Connection Type: | Direct |
| Encryption: | Disabled |

*Figure 6.4. DCV Console About window*

## Advanced settings

By clicking on the Advanced expander you can access more settings and check network connection statistics.

Figure 6.5, “DCV Console advanced window” shows the advanced settings.

## Show frames per second

If checked shows the frames per second received by the end station. Each 3D window will display an overlay FPS number in the bottom right corner.

## Bandwidth usage

DCV Console provides statistics on the network link condition between the server and the end station.

A sliding graph shows the network bandwidth usage by DCV in the last minute. The red line shows the instant value, the blue line is the average in the last minute.

In addition to the current value and the average value, the graph box also shows the peak bandwidth from the time the end station connected.

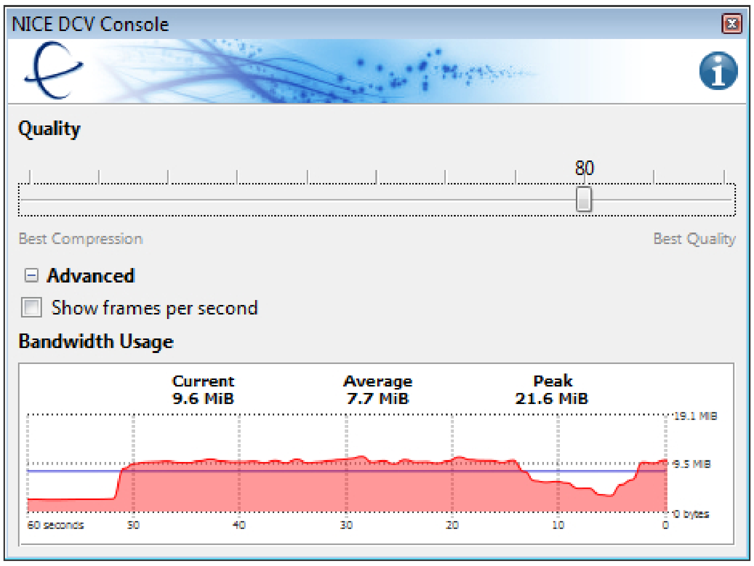
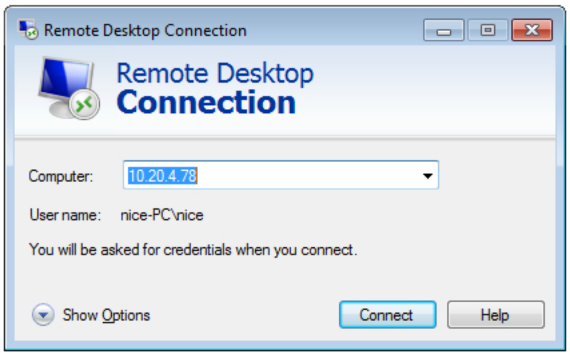


*Figure 6.5. DCV Console advanced window*

## Using DCV in a Remote Desktop Connection

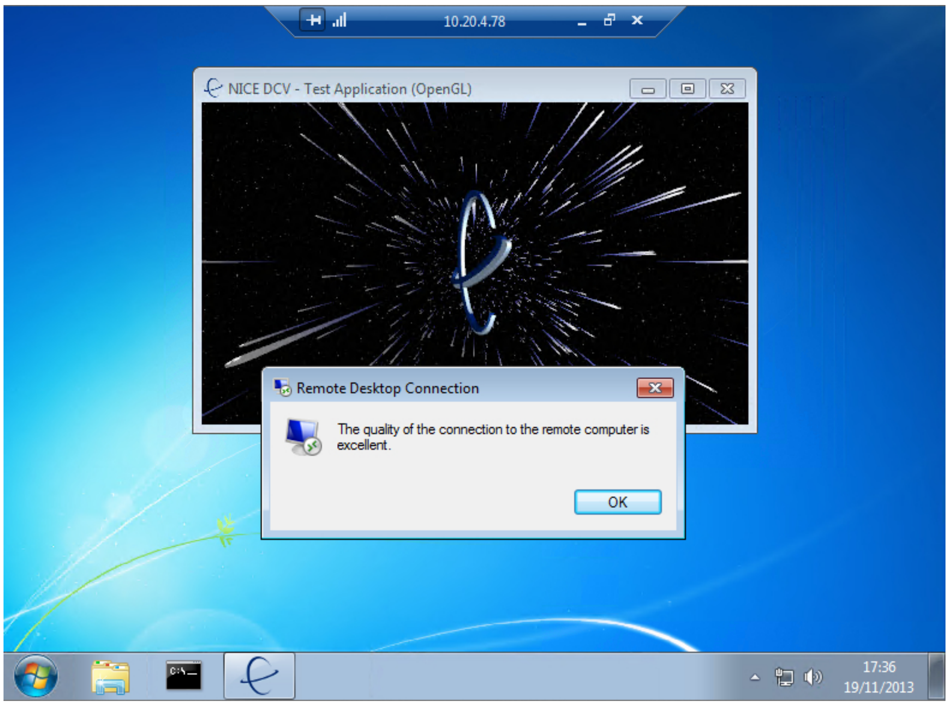
To start using DCV in a Remote Desktop Connection you just need to use a Remote Desktop Client and connect to the remote Application Host. Figure 6.6, "Remote Desktop Client connection dialog" shows the default connection dialog of the Remote Desktop Client.



Computer: 10.20.4.78
User name: nice-PC\nice
You will be asked for credentials when you connect.

*Figure 6.6. Remote Desktop Client connection dialog*

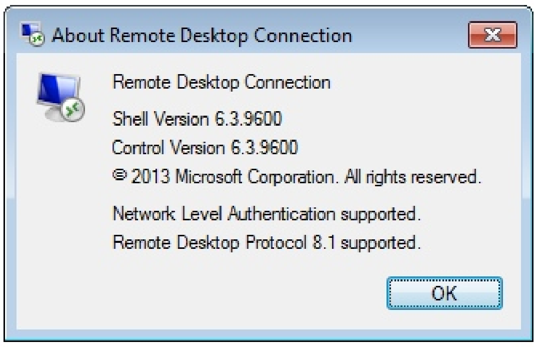
When connected to a remote desktop you can launch the DCV Test application to check Desktop Cloud Visualization is working.


10.20.4.78
NICE DCV - Test Application (OpenGL)
Remote Desktop Connection
OK
17:36
19/11/2013

The quality of the connection to the remote computer is excellent.

*Figure 6.7. DCV Test application running in a Remote Desktop Connection*

You can check the RD Client version and supported RD Protocol by opening the About dialog.



Remote Desktop Connection

Shell Version 6.3.9600

Control Version 6.3.9600

© 2013 Microsoft Corporation. All rights reserved.

Network Level Authentication supported.

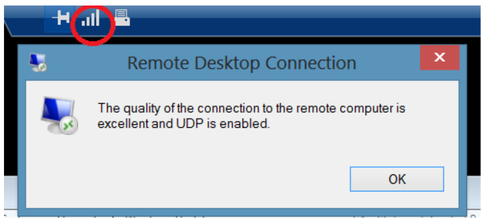
Remote Desktop Protocol 8.1 supported.

*Figure 6.8. Remote Desktop Client about dialog*

For optimal performance, both the client and server must support Remote Desktop Protocol 8.0 or above.

If RDP 8.0 is enabled, the connection quality button will be displayed in the connection bar.

Click the connection quality button to open an information dialog box that resembles the following. See Figure 6.9, “Check RDP 8.0 is enabled”.



The quality of the connection to the remote computer is excellent and UDP is enabled.

*Figure 6.9. Check RDP 8.0 is enabled*

## Note

When using DCV inside a Remote Desktop Connection the delivery of 3D images is delegated to RDP while Desktop Cloud Visualization takes care of the GPU virtualization for OpenGL calls.

When using Remote Desktop Client, the DCV Console is not available and Quality settings is delegated to the RDP configuration.

Remote Desktop Client is available on many platforms. Figure 6.10, “Microsoft Remote Desktop for Android” shows DCV Test application running inside a Microsoft Remote Desktop for Android.

Check on Microsoft web site for more information on how to download the appropriate client for your platform.

*Figure 6.10. Microsoft Remote Desktop for Android*

## Using DCV on an Application host

On an application host working with DCV revolves around the following activities:

- administering DCV (reserved to users with administrator privileges);
- enabling and configuring a DCV session;
- executing OpenGL applications.

To achieve these tasks DCV provides both command line and GUI utilities.

### Command Line Tools

`dcvadmin.exe (Windows with External Rendering Server)/ dcvadmin (Linux)`

The `dcvadmin` command allows an administrator to manage DCV on a system. By running `dcvadmin enable` the DCV libraries will be deployed on the system while issuing `dcvadmin disable` will undeploy them. The DCV libraries are deployed automatically by the installer, so the use of the `dcvadmin` command is required only when upgrading the graphic drivers or when performing system administration tasks. The `dcvadmin status` command allows a user to inspect if DCV components are deployed on the system.

`dcv.exe (Windows with External Rendering Server) / dcv (Linux)`

The `dcv` command allows a user to switch on and off DCV for the current desktop session. When `dcv on` is issued, DCV becomes active and all OpenGL applications subsequently started will make use of DCV. In a similar way DCV can be deactivated with the `dcv off` command. OpenGL applications making already use of DCV will not be impacted by the `dcv off` command. Finally `dcv status` provides information about the activation state of DCV on the current desktop.

On Linux the `dcv` command optionally accepts a VNC X display number as an argument so that the action is applied to the specified display instead of the current one. For instance `dcv on :2.0` will activate DCV on the VNC X display :2.0.

`dcvtest.exe (Windows)/ dcvtest (Linux)`

The `dcvtest` command starts a simple test application showing NICE and DCV logos rotating over a star field.

Press spacebar to pause the animation.

`dcvscrun (Linux)`

The `dcvscrun` *(DCV SmartCard Run)* command allows to run applications which use the standard PC/SC interface so that they can use the DCV Smart Card Remotization.

To launch an application with `dcvscrun` type:

```
dcvscrun <app> [app arguments]
```

For more information please refer to the the section called “Smart Card Remotization”.

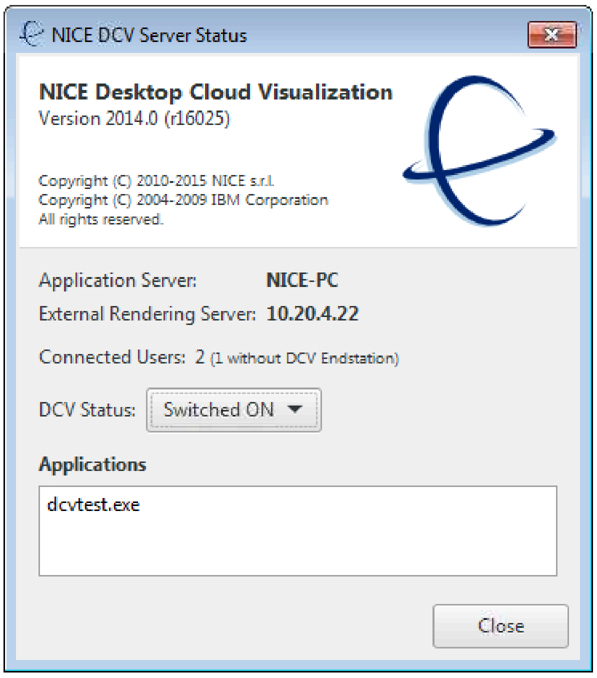
`dcvstartx (Linux)`

The `dcvstartx` *(DCV Start X)* command allows to start a minimal X11 server showing the DCV logo. It provides a minimal setup, preventing user logins on the graphic console and takes care of enabling access to local users by calling the `dcvxgrantaccess` command.

For more information please refer to the the section called “Grant Local Connections to the 3D Rendering Display”.

## The DCV Config utility

The DCV Config utility provides a graphical interface to check DCV status. On Linux and on Windows with External Rendering Server it allows to switch DCV on and off for the current session and to list the applications remotized through DCV.


NICE DCV Server Status

NICE Desktop Cloud Visualization
Version 2014.0 (r16025)

Copyright (C) 2010-2015 NICE s.r.l.
Copyright (C) 2004-2009 IBM Corporation
All rights reserved.

Application Server: NICE-PC
External Rendering Server: 10.20.4.22

Connected Users: 2 (1 without DCV Endstation)

DCV Status: Switched ON

Applications

dcvtest.exe

*Figure 6.11. DCV Config showing External Rendering Server*

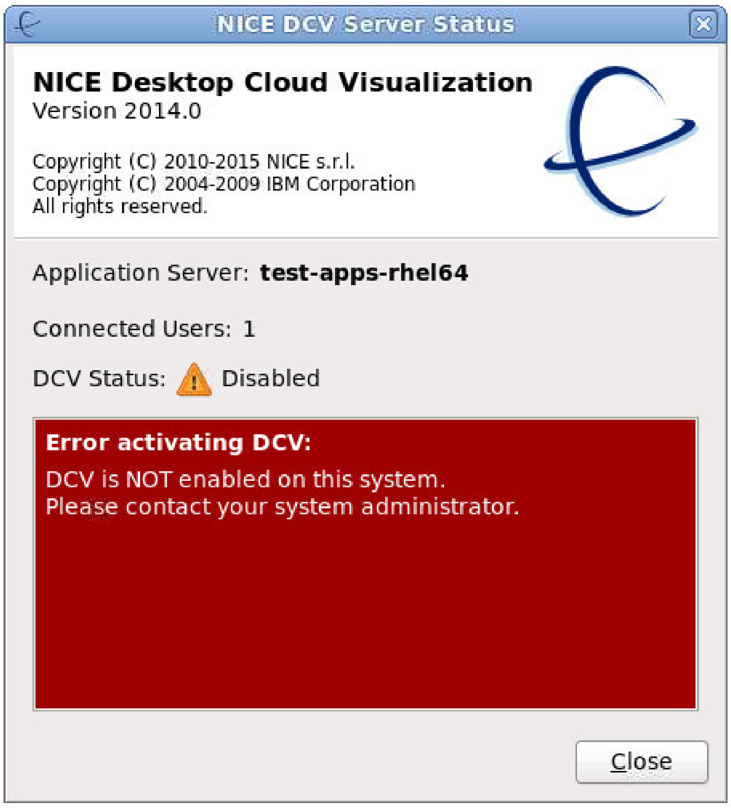
It also reports the number of connected users and how many of them are not using a DCV End Station.

On Windows the DCV Config utility starts up automatically at login if DCV is enabled. On Linux the DCV Config utility is started automatically the first time `dcv on` is executed (this usually occurs while executing the `xstartup` file).

The DCV Config utility is accessible by clicking on the DCV icon in the system tray area. On Linux, in case the tray area is not available, the DCV Config utility will show its main window at startup.

To avoid the DCV Config utility to be shown at session startup and to be added to the tray area, use `dcv on --nowin`.

The `DCV Server` shows the hostname of the server DCV is running on, and on Windows with External Rendering Server, the default External Rendering Server.


NICE DCV Server Status

NICE Desktop Cloud Visualization
Version 2014.0

Copyright (C) 2010-2015 NICE s.r.l.
Copyright (C) 2004-2009 IBM Corporation
All rights reserved.

Application Server: **test-apps-rhel64**

Connected Users: 1

DCV Status: ⚠ Disabled

**Error activating DCV:**
DCV is NOT enabled on this system.
Please contact your system administrator.

*Figure 6.12. DCV Config diagnostic messages*

The DCV Config utility also provides the following diagnostic information in case of problems:

- DCV is installed but not enabled on the system (see Figure 6.12, “DCV Config diagnostic messages”);
- DCV cannot find a valid license;
- DCV cannot start because a system problem occurred.

## Running an Application with DCV on Linux

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

### Display Isolation Mode

To run an application with DCV in *Display Isolation Mode*, a standalone VNC server must be started on the application host:

1. log in to the application host and run the following command:

```
vncserver
```

   for more information on how to use the `vncserver` command, refer to

```
man vncserver
```

.

The VNC server will execute the `xstartup` file to setup an isolated desktop session.

The DCV installer adds automatically the `dcv on` command at the end of the `/etc/vnc/xstartup` script so that the new session is ready to use OpenGL applications with DCV.

Once the VNC server is up and running you can connect to it using the `vncviewer` and then to start an application you can perform the same steps described for the the section called "Native Display Mode (System Console)".

### Native Display Mode (System Console)

To run an application with DCV in *Native Display mode*, the VNC module must be loaded in the X server, so that it is possible to connect to the host using the `vncviewer` client.

Once the VNC session is active, to start an OpenGL application making use of DCV, you can follow these steps:

1. check in the DCV Config that DCV is active. Click on the `Switch on` button if DCV is not already active;
2. start the OpenGL application;
3. after few seconds the new application will appear in the DCV Config application list.

## Running an Application with DCV Rendering Server

To run an application with DCV Rendering Server, ensure the RealVNC Visualization Edition Server is running on the application host machine, so that it is possible to connect to the host using the `vncviewer` client.

Once the VNC session is active, to start an OpenGL application making use of DCV, you can follow these steps:

1. check in the DCV Config that DCV is active. Click on the `Switch on` button if DCV is not already active;
2. start the OpenGL application;
3. after few seconds the new application will appear in the DCV Config application list.

# 7 Diagnostics

## Verifying Installation

When the installation procedure is complete, you can verify that DCV is properly installed.

Desktop Cloud Visualization includes a simple OpenGL/DirectX application that can be used to test whether DCV is working correctly.

### Verification of DCV Installation on Linux

To verify DCV installation, follow these steps:

1. on the application host: launch a new `vncserver` process

2. on the end station: start the DCV End Station and connect to the appropriate display on the application host.

   You should now see the remote desktop

3. from the remote desktop, check the DCV Config tray icon and click on it

4. check DCV is switched on. If needed use the DCV Status control to turn it on

5. run the Desktop Cloud Visualization test application on the application host using the `dcvtest` command

6. the application should start with the following window title: "NICE DCV - Test Application"

   an item (dcvtest.bin) should appear in the DCV Config Applications area

   if the graphics are displayed as in Figure 7.1, “Linux DCV Test Application” then the installation on the application host was successful

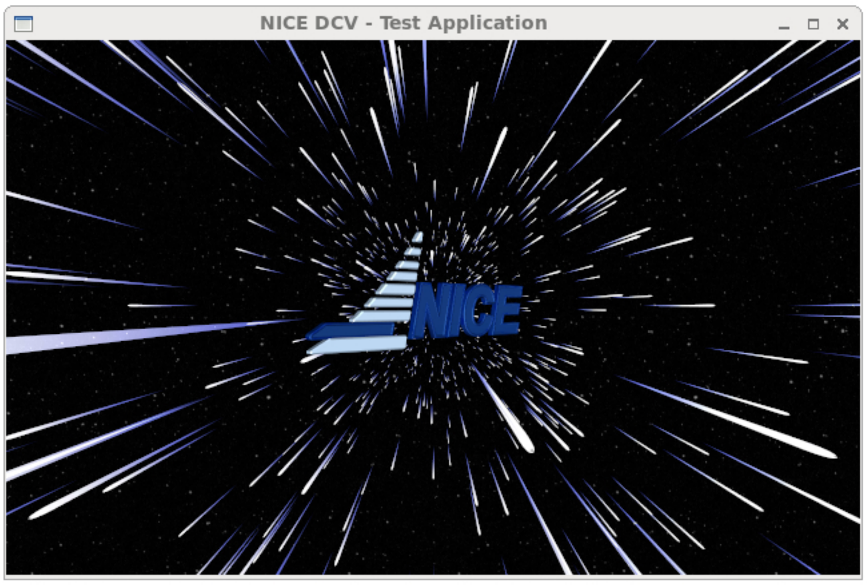


*Figure 7.1. Linux DCV Test Application*

## Verification of DCV installation on Windows

To verify DCV installation, follow these steps:

1. on the application host: check the VNC server and `dcvd` processes are running
2. on the end station: start DCV End Station and connect to the application host

   You should now see the remote desktop;
3. from the remote desktop, run the Desktop Cloud Visualization test application:

   click `Start > All Programs > NICE > DCV > DCV Test Application`
4. The application should start with the following window title: "NICE DCV - Test Application (OpenGL)"

   If the graphics are displayed as in Figure 7.2, “Windows with External Rendering Server Test Application” then the installation on the application host was successful.

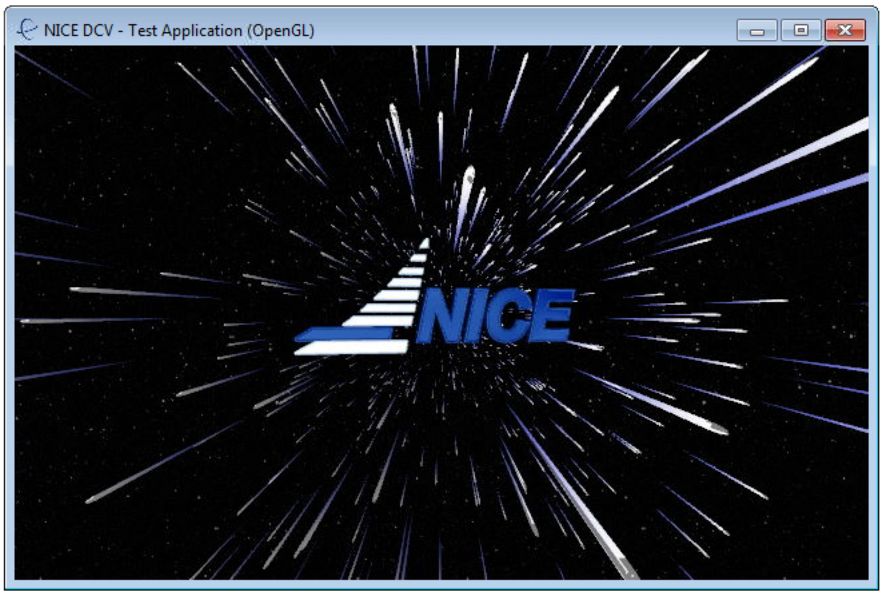


*Figure 7.2. Windows with External Rendering Server Test Application*

# DCV Problems

This chapter describes how to detect and solve problems that might occur with DCV.

Starting from version 2013, DCV for Linux ships with a useful diagnostic script able to detect the most common problems and provide suggestions on how to fix them.

### Note

In case you are experiencing a problem with Linux, run `dcvdiag` as `root` before contacting NICE support.

If you cannot determine the cause of failure, request the assistance of the NICE support contacting <`helpdesk@nice-software.com`>.

Provide the log file of `dcvdiag` in your support requests.

If you encounter any problems using DCV, 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. If these actions do not solve the problem, follow local problem-reporting procedures.

1. Use only a single end station.
2. If the Windows end station does not display the 3D graphics, ensure that the system `PATH` environment variable on the end station contains the correct DCV binary directories.

   After an end station installation a reboot may be required for the changes to the system `PATH` to take effect.
3. Run the installation verification procedure. See the section called “Verifying Installation” .

## Cannot Init Local Display

When launching an OpenGL application on Linux application hosts it does not start and you see the following error:

```
Xlib: connection to ":0.0" refused by server
Xlib: No protocol specified

DCV RVN: error, cannot init local display.
```

DCV uses a 3D hardware accelerated X11 display, the "3D rendering display", to render OpenGL accelerated graphics.

Usually the 3D rendering display is the first display, `:0`, but it may change for different reasons, for example in case multiple 3D video cards are installed.

To ensure DCV can render 3D graphics:

1. the OpenGL application must know the 3D rendering display to connect to;
2. the 3D rendering display must grant X connections from the local machine to any user.

### Specify the 3D Rendering Display to Use

By default DCV will use the first `:0.0` display as 3D rendering display.

In case the 3D rendering display is different it is possible to explicitly specify the display to use. You can permanently configure the display in the `dcv.conf` file (see Appendix A, *`dcv.conf`*) or, if you want to change the display just for the execution of an application, you can use the RVN_LOCAL_DISPLAY environment variable.

Example: to make the DCV Test application use the second screen on the first display, use the following line:

```
RVN_LOCAL_DISPLAY=:0.1 dcvtest
```

### Grant Local Connections to the 3D Rendering Display

DCV libraries run inside the 3D applications launched by the users and redirect 3D calls to the accelerated display. So the application processes run by the users must be able to access the 3D accelerated X display.

DCV provides a script, `/opt/nice/dcv/bin/dcvxgrantaccess`, whose purpose is to grant access to the 3D accelerated X display configured in `dcv.conf`. By default `dcvxgrantaccess` grants access to all local users of the system.

It is important to understand the security risks implied by this configuration. Users having access to a display could send or receive keystrokes and grab images from that display. Therefore it is important to never use the 3D accelerated X Server to login in, expecially as `root`.

It is however possible to restrict the access to a subset of users, providing a custom implementation of `dcvxgrantaccess`.

To let the 3D rendering display accept X connections on a production system we suggest one of the following methods:

1. the system starts at `runlevel 3` (text mode).

   Configure `/opt/nice/dcv/bin/dcvstartx` as a startup script into `/etc/init.d`.

   The `dcvstartx` script will take care of starting a minimal X server showing the DCV logo and calling `/opt/nice/dcv/bin/dcvxgrantaccess`.

2. the system starts at `runlevel 5` (graphics mode with GDM or other Display Manager for login).

   When running `dcvadmin enable`, DCV automatically searches for well-known X or display manager startup scripts and adds a call to `/opt/nice/dcv/bin/dcvxgrantaccess`.

   After running `dcvadmin enable` the first time, it is required to stop and start the X server to apply the changes, for instance by switching to runlevel 3 and then go back to runlevel 5.

#### Note

When using this method the X display on the console screen will continue to show the GDM/KDM login screen but it should NOT be used. Logging-in and out cause the X process to be restarted. All applications using DCV, that actually render 3D on that display, will loose their 3D panel or they can even crash.

In case the dcvadmin script is not able to detect the X or Desktop Manager start up script, you need to manually find the script according to you distribution and setup and add the following command at the beginning:

```
/opt/nice/dcv/bin/dcvxgrantaccess
```

The appropriate script location may change depending on the Linux distribution or the installed Display Manager.

For example on a Red Hat Enterprise Linux 5 using the script is `/etc/gdm/Init/Default` while on SuSE it is `/etc/opt/gnome/gdm/Init/Default`.

If your system is using KDM the suggested script is `Xsetup` usually located under `/usr/share/config/kdm`, `/etc/kde3/kdm` or `/etc/kde4/kdm`.

Please refer to your Linux distribution documentation for further information.

## VNC Problems

### Font Path Problems

When launching a `vncserver` process you may get the following error:

```
VNC Server Visualization Edition VE4.6.x (rXXXXX)
Copyright (C) 2002-2009 RealVNC Ltd.
See http://www.realvnc.com for information on VNC.
[...]
Could not init font path element /usr/X11R6/lib/X11/fonts/100dpi/ [...]

Fatal server error:
could not open default font 'fixed'
```

New versions of X servers expect fonts to be in a different location so you must specify the correct font paths in the `/etc/vnc/config`.

This is an example configuration from a Red Hat Enterprise Linux 5 installation:

```
# Default vncserver configuration. See the vncserver man page for details.

-PasswordFile $HOME/.vnc/passwd          # Location of VncAuth password
-desktop "$HOSTNAME:$DISPLAYNUM ($USER)" # Desktop name
-pn                                      # Continue even if standard
                                         # ports fail
-httpd <inline>                          # Serve Java viewer inline

# Default RGB database file
-co "/usr/share/X11/rgb"

# Updated font paths for Red Hat Enterprise Linux 5
-fp "/usr/share/X11/fonts/misc/:unscaled, /usr/share/X11/fonts/100dpi/:unscal
ed, /usr/share/X11/fonts/75dpi/:unscaled, /usr/share/X11/fonts/misc/, /usr/sh
are/X11/fonts/Type1/, /usr/share/X11/fonts/100dpi/, /usr/share/X11/fonts/75dp
i/, /usr/share/X11/fonts/TTF/"
```

## Mirror Driver

On Windows with External Rendering Server, RealVNC Server should use VNC Mirror Driver for optimal performances.

In case RealVNC Server installation is done remotely via Remote Desktop Connection, RealVNC is unable to install the VNC Mirror Driver.

To install RealVNC Server remotely on Windows proceed in this way:

- connect via Remote Desktop Connection and install DCV which in turn installs the RealVNC Server
- VNC installation will warn with the following message:

  ```
  VNC Mirror Driver can only be installed on the console or
  over a VNC connection.
  Please reconnect via VNC and install the VNC Mirror Driver manually.
  ```

  Ignore the message and continue the installation
- disconnect from RDP and connect via VNC
- right click on the VNC tray icon, select `Status...`; the VNC Server Status window will display a warning: "VNC Mirror Driver is not available [ details ]". Click on the "[ details ]" link
- a tooltip will popup explaining the issue: "[...] If you have not installed it yet, please click here to do so. [...]". Click on the "click here" link
- the mirror driver will be installed, the screen may be blocked during the installation. Disconnect (you should be disconnected automatically after a while)
- reconnect and check the VNC Server Status window. It will display a green status: "VNC Server is working normally in Service-Mode"

**Note**

In a normal Windows installation (without External Rendering Server) the Mirror Driver is not installed and must not be used.

## Linux desktop environment cannot be loaded

Some Linux desktop environments, like GNOME 3 (the default desktop environment used by Red Hat Enterprise Linux 7.x), do not work under DCV Server because they require capabilities that are not available in the bundled X server (Xvnc). This is also true for the "2D" or "fallback" modes, which used to work in previous releases of the same desktop environments. In such cases, we recommend installing and using an alternative desktop environment like KDE Plasma Desktop, Xfce (with the Xvnc RENDER extension disabled) or LXDE.

For more details about this problem and how to configure the VNC server startup file to use a different desktop environment, please refer to the RealVNC Knowledge base article VNC Server in Virtual Mode appears to hang or you see a grey screen because a desktop environment cannot be loaded

Distributions which use GNOME 2 are also known to work well (for example Red Hat Enterprise Linux 6).

## Blank the screen not working

RealVNC Server for Windows provides the option "Blank the screen while VNC Viewers are connected" under the Desktop tab in the Privacy section.

This option is known not to work with DCV.

DCV makes use of NVIDIA NvFBC fast-readback technology and this is not compatible with screen blanking.

# X Server Display Problems

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

- `/var/log/XFree86.0.log` or `/var/log/Xorg.0.log`
  - This file contains information about the X Server.
- **xdpyinfo**
  - This display utility provides information for X, including whether accelerated 3D graphics are enabled and whether required modules are loaded.

    To check whether 3D acceleration is enabled on the `:0` display run:

    ```
    # DISPLAY=:0  xdpyinfo | grep GLX
    ```

    The output should contain the "GLX" and "NV-GLX" words.

- **xhost**
  - Shows information about which machines are allowed to display on the local X Server.

## No Monitor Attached to the Video Adapter

In case there is no monitor attached to the NVIDIA video adapters some X Servers may refuse to start.

To solve the problem modify the `Device` section in the xorg.conf: add the `UseDisplayDevice` option and set it to `none`.

Example:

```
Section "Device"
  Identifier "Device0"
  Driver     "nvidia"
  Option     "UseDisplayDevice" "none"
EndSection
```

## Multiple Video Cards

In case there are multiple video adapters the X Server must be configured to use them and with the correct drivers.

For each NVIDIA card a `Device` section (specifying the appropriate `Driver` and `BusID` values) and a `Screen` section (using the device) must be present. An appropriate `ServerLayout` section must also be present to arrange the screens.

Using the `nvidia-xconfig` NVIDIA utility, it is possible to automatically create an xorg.conf file including all available NVIDIA GPUs:

```
# nvidia-xconfig --enable-all-gpus -o autodetected.xconf.conf
```

After creating the new configuration file, check its correctness before using it.

Below an extract of an xorg.conf showing the Device, Screen, ServerLayout sections for a dual card server. The UseDisplayDevice has been manually added.

```
Section "ServerLayout"
  Identifier     "Default Layout"
  Screen      0  "Screen0"
  Screen      1  "Screen1" RightOf "Screen0"
  InputDevice    "Mouse0" "CorePointer"
  InputDevice    "Keyboard0" "CoreKeyboard"
EndSection

Section "Device"
  Identifier "Device0"
  Driver     "nvidia"
  VendorName "NVIDIA Corporation"
  BoardName  "Tesla M2070"
  BusID      "PCI:6:0:0"
  Option     "UseDisplayDevice" "none"
EndSection

Section "Device"
  Identifier "Device1"
  Driver     "nvidia"
  VendorName "NVIDIA Corporation"
  BoardName  "Tesla M2070"
  BusID      "PCI:7:0:0"
  Option     "UseDisplayDevice" "none"
EndSection

Section "Screen"
  Identifier     "Screen0"
  Device         "Device0"
  DefaultDepth    24
  SubSection     "Display"
    Depth       24
  EndSubSection
EndSection

Section "Screen"
  Identifier     "Screen1"
  Device         "Device1"
  Monitor        "Monitor1"
  DefaultDepth    24
  SubSection     "Display"
      Depth       24
   EndSubSection
EndSection
```

The `BusID` value can be obtained on modern NVIDIA drivers using:

```
# nvidia-xconfig --query-gpu-info
```

In case the nvidia-xconfig is not available on your system, you can use the `lspci` command:

```
# lspci
...
06:00.0 VGA compatible controller: nVidia Corporation GT200GL [...]
07:00.0 3D controller: nVidia Corporation GT200 [...]
```

**Note**

Note the output by lspci is in hexadecimal format and must be converted before using in xorg.conf. For example, `25:00.0` must be converted to `PCI:37:0:0`

## NVIDIA Display Problems

If you are having NVIDIA display problems on Linux, go through the following check-list and take the suggested corrective actions if needed:

- NVIDIA drivers conflict with the open source *nouveau* drivers shipped with some modern Linux distributions. Check the *nouveau* drivers are not loaded with the following command:

  ```
  lsmod | grep nouveau
  ```

  The output must be empty.

  To prevent *nouveau* drivers to be loaded from the Linux Kernel you may need to blacklist them both in the bootloader and the modprobe configuration. Please refer to the documentation from your Linux distribution for specific information.

  On RHEL 6 this is the suggested way to disable *nouveau* drivers:

  - In `/boot/grub/grub.conf` append the following two kernel options: `rdblacklist=nouveau nouveau.modeset=0` as in this example:

    ```
    title CentOS
      root (hd0,0)
      kernel /vmlinuz-... rdblacklist=nouveau nouveau.modeset=0
      initrd /initramfs-...
    ```

  - In `/etc/modprobe.d/blacklist` append a new line at the end as in this example:

    ```
    ...
    # nouveau driver
    blacklist nouveau
    ```

  - Reboot the system, check *nouveau* drivers are not loaded and install NVIDIA drivers.

- if you update the Linux Kernel, you will need to re-install display drivers;

- if you install or upgrade display drivers, you will need to reboot the system and run again the `dcvadmin enable` adminstrator level command;

- check the `nvidia` kernel module is correctly installed executing:

  ```
  lsmod | grep nvidia
  ```

The output should contain a line like the following:

```
nvidia              11704102  40
```

- check the the PCI Bus is reporting the display card executing:

  ```
  lspci
  ```

  The output should contain a line like the following:

  ```
  06:00.0 VGA compatible controller: nVidia Corporation Device 06dd (rev a3)
  ```

- check the availability of `nvidia-smi` program and execute:

  ```
  nvidia-smi -a
  ```

  In case `nvidia-smi` is not available on your installation, try to execute:

  ```
  cat /proc/driver/nvidia/version
  NVRM version: NVIDIA UNIX x86_64 Kernel Module  340.46
  GCC version:  gcc version 4.8.2 20140120 (Red Hat 4.8.2-16) (GCC)
  ```

  and verify that the correct driver version is installed, and reinstall it if necessary.

  Depending on the version, additional information can be retrieved with:

  ```
  cat /proc/driver/nvidia/cards/[CARD NUMBER]
  ```

  or

  ```
  cat /proc/driver/nvidia/gpus/[GPU ADDRESS]/information
  ```

  Verify that the reported card is a supported display adapter. If the adapter is not supported, install a supported card and its associated driver.

- if these actions do not solve the problem, follow local problem-reporting procedures.

## License Problems

On Linux, when trying to turn on DCV via `dcv` command you see an error similar to the following:

```
dcv on
      DCV cannot be switched ON
      <additional error specific messages>
```

Check the license file is present, its expiration and the host id matches the one returned by the `rlmutil -hostid`.

For more information set `RLM_EXTENDED_ERROR_MESSAGES=1` in the environment and try to turn on DCV again.

In case the problem persists, contact <helpdesk@nice-software.com> or NICE Sales providing the output from the `dcv on` command.

## Performance Issues

In case you experience performance issue running DCV go through the following check-list and take the suggested corrective actions if needed:

- check the behavior of the same OpenGL application when running directly on the server machine console;

  **Important**

  If the application is not performing as expected on the console contact the support from the application or hardware vendor.

- on Linux servers, check the GPU is configured for performance running as administrator from a graphic session:

  ```
  dcv-nvidia-settings
  ```

  For each GPU check on the *PowerMizer* page the *Performance Level* is the greatest possible. To enable this setting you can set the *Preferred Mode* to *Prefer Maximum Performance* from the NVIDIA settings GUI but this setting will be lost after a reboot.

- on Windows with External Rendering Server, check the RealVNC Mirror Driver is correctly installed and working;

- on Linux servers check the CPU performance configuration. Refer to you Linux distribution documentation for more information.

  On Red Hat Enterprise Linux 5/6 you can check the current scaling governor for each core running:

  ```
  cat /sys/devices/system/cpu/cpuX/cpufreq/scaling_governor
  ```

  The output should be: `performance`

  To change the current value edit `/etc/sysconfig/cpuspeed` and set the `GOVERNOR` param to `performance`:

  ```
  GOVERNOR=performance
  ```

  Then restart `cpuspeed` with:

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

  and check the changes are applied with:

  ```
  # /etc/init.d/cpuspeed status
  ```

  Check `cpuspeed` is started at boot in runlevel 3 and 5 with:

```
# chkconfig --list cpuspeed
```

## DCV Config reports client without DCV End Station

In case the DCV End Station is not correctly installed or the client is not using the DCV End Station but a different VNC Viewer, the DCV Config tool reports at least one of the "Connected Users" is "without DCV Endstation". DCV cannot optimize the 3D images for such clients.

If the client supports the DCV End Station follow Chapter 4, *Installing DCV End Station* and reinstall it.

## Maximum number of FPS

On Linux and Windows DCV optimizes bandwidth usage by limiting the maximum number of frames per second sent over the network. The default FPS limit is set to 25 which is usually fine for the most use cases but it can be changed if required.

Please refer to the usage of the section called “`target_fps`” for more information.

## Performance Issues with External 3D Rendering

Additional considerations apply when running DCV with External 3D Rendering:

- check network latency between rendering host and application host running the `ping` command. To ensure good performance the roundtrip time reported by the `ping` command should not be above 0.3 milliseconds;

- check network bandwidth between the rendering host and the application host in both directions. To ensure good performance the bandwidth should not be below 1-4 Gbit/s.

  To measure the actual bandwidth between the rendering host and the application host you can use the `iperf` (`http://sourceforge.net/projects/iperf/`) tool which is freely available for both Linux and Windows. You can test the upload bandwidth from the application host to the rendering host running the following commands, on the rendering server:

  ```
  iperf -s -B <rendering host IP address on host-only network>
  ```

  on the application server:

  ```
  iperf -c <rendering host IP address on host-only network>
  ```

- check general virtual application host performance (vCPU, vDisk, RAM). Refer to the hypervisor manual for VM performance tuning.

  As reference the Windows guest should have Windows Experience Index values in line with or better than:

  - Processor: 7

  - Memory (RAM): 7

- Primary Hard Disk: 6

# Appendix A. `dcv.conf`

You can customize DCV via the `dcv.conf` configuration file located in the `conf` directory under the DCV installation.

This file specifies rules for different DCV components.

It is a text file with a basic structure composed of "sections" and "properties" based on the INI format.

The section name appears on a line by itself, in square brackets. All properties after the section declaration are associated with that section. There is no explicit "end of section" delimiter; sections end at the next section declaration, or the end of the file. Sections can not be nested.

Each property has a key and a value, delimited by an equals sign. The key appears to the left of the equals sign. Properties may be grouped into arbitrarily named sections.

Lines starting with a hash mark or semi-colon and empty lines are ignored.

Each entry in dcv.conf has one of the following forms:

- `[section]`
- `KEY=VALUE`

## `[Application]`

List of `[Application]` settings.

### `external_rendering_server`

#### Syntax

`external_rendering_server` = *host*[*:port*]

#### Description

Specifies the hostname or IP address followed by the port of the external rendering server that the OpenGL applications must use to delegate 3D rendering.
The value of the `external_rendering_server` parameter can be overridden by setting the environment variable `DCV_RENDERING_SERVER`.

#### Valid Values

Hostname or IP address followed by the port of a DCV rendering server. Port may be omitted.

#### Default Value

No external rendering server. In case just host is specified the default port is 2007.

**Note**

The communication between OpenGL applications and the DCV rendering server must go through a high bandwidth/low latency network connection.

## `local_display_list`

### Syntax

`local_display_list` = :*display*[.*screen*] ...

### Description

List of local 3D accelerated X displays and screens used by DCV for OpenGL rendering.
The value of the `local_display_list` parameter can be overridden by setting the environment variable `RVN_LOCAL_DISPLAY`.

### Valid Values

Space separated list of local 3D accelerated X displays.

### Default Value

`:0.0`

## `local_multisample`

### Syntax

`local_multisample` = *integer*

### Description

Specifies the number of samples to request. Use 0 to turn off multisampling, use -1 to choose the highest supported value.
The value of the `local_multisample` parameter can be overridden by setting the environment variable `RVN_LOCAL_MULTISAMPLE`.

### Valid Values

- `-1`: highest supported multisampling;
- `0`: turn multisampling off;
- `greater than 0`: the multisampling number to use.

### Default Value

`0`

**Note**
Increasing the number of multisampling requires more computing power and decreases OpenGL rendering performances.

## `print_dcv_version`

### Syntax

`print_dcv_version` = *boolean*

### Description

Specifies if DCV should print a message with the DCV version when an OpenGL application starts.

The value of the `print_dcv_version` parameter can be overridden by setting the environment variable `DCV_PRINT_VERSION`.

### Valid Values

true | false

### Default Value

`false`

## `target_fps`

### Syntax

`target_fps` = *integer*

### Description

Specifies the frame rate at which frames are grabbed and sent on the network. The Target FPS can be reached or not depending on:

- Whether or not the application is generating enough frames on server side;
- The time spent to generate compressed the frames;
- The available bandwidth permits to send the frames;
- The end station is able to decompressed the frames.

The value of the `target_fps` parameter can be overridden by setting the environment variable `RVN_TARGET_FPS`.

### Valid Values

- `-1, 0`: disables the limit;
- `greater than 0`: sets the limit to that number.

### Default Value

`25`

## `use_nvidia_grid`

### Syntax

`use_nvidia_grid` = *boolean*

### Description

Specifies to use the NVIDIA GRID technology.
The value of the `use_nvidia_grid` parameter can be overridden by setting the environment variable `DCV_USE_NVIDIA_GRID`.

### Valid Values

true | false

### Default Value

`true`

## use_sample_queries

### Syntax
use_sample_queries = *boolean*

### Description
Specifies whether DCV should use GL sample queries or not.
The value of the use_sample_queries parameter can be overridden by setting the environment variable RVN_USE_SAMPLE_QUERIES.

### Valid Values
true | false

### Default Value
true

## [Endstation]

List of [Endstation] settings.

## console_shortcut

### Syntax
console_shortcut = should be something like "<Control><Shift>F9"

### Description
Sets the shortcut to launch the console
The value of the console_shortcut parameter can be overridden by setting the environment variable RVN_CONSOLE_SHORTCUT.

### Valid Values
Valid modifiers are Control, Shift and Alt. These values must be contained between < and > characters.

### Default Value
<Control><Shift>F9

**Note**
See that in order to set these characters in an environment varaible you need to escape them with ^ on Windows and with \ on Linux.

## enable_console

### Syntax
enable_console = *boolean*

### Description
Specifies whether to load and use the client-side console.

The value of the `enable_console` parameter can be overridden by setting the environment variable `DCV_ENDSTATION_ENABLE_CONSOLE`.

#### Valid Values

true | false

#### Default Value

`true`

### `encryption`

#### Syntax

`encryption` = server | alwaysmaximum | alwayson| preferon | preferoff

#### Description

The endstation encryption mode.
The value of the `encryption` parameter can be overridden by setting the environment variable `DCV_ENDSTATION_ENCRYPTION`.

#### Valid Values

`server` uses the server's encryption preferences `alwaysmaximum` to use always encryption with a 256 bits encryption key `alwayson` to connect to servers that use at least 128-bit encrypted connections `preferon` to use 128 bits encryption if the endstation allows the server to decide or if sets preferon `preferoff` to not use encryption

#### Default Value

`server`

### `proxy_port`

#### Syntax

`proxy_port` = *integer*

#### Description

Specify the TCP port number of the HTTP or SOCKS5 proxy server that the DCV end station must use to connect to the DCV server.
The value of the `proxy_port` parameter can be overridden by setting the environment variable `DCV_PROXY_PORT`.

#### Valid Values

A valid TCP port number.

#### Default Value

8123

### `proxy_server`

#### Syntax

`proxy_server` = *hostname* | *IP address*

### Description

Specify the hostname or IP address of the HTTP or SOCKS5 proxy server that the DCV endstation must use to connect to the DCV server.
The value of the `proxy_server` parameter can be overridden by setting the environment variable `DCV_PROXY_SERVER`.

### Valid Values

Hostname or IP address of a HTTP or SOCKS5 proxy server. Empty string to directly connect to the DCV server without using any proxy server.

### Default Value

No proxy server.

> **Note**
> Proxy servers that require authentication are not supported. This may be the subject of a future release.

## proxy_type

### Syntax

`proxy_type` = `HTTP | SOCKS5`

### Description

Specify the type of proxy server that the DCV end station must use to connect to the DCV server.
The value of the `proxy_type` parameter can be overridden by setting the environment variable `DCV_PROXY_TYPE`.

### Valid Values

`HTTP` to use a HTTP proxy or `SOCKS5` to use a SOCKS5 proxy

### Default Value

`HTTP`

## tip_timeout

### Syntax

`tip_timeout` = milliseconds

### Description

Sets the time the console shortcut tip will be present.
The value of the `tip_timeout` parameter can be overridden by setting the environment variable `DCV_ENDSTATION_TIP_TIMEOUT`.

### Valid Values

Any positive integer.

### Default Value

`10000`

# [License]

List of [License] settings.

## license_file

### Syntax

license_file = *filename* | *port@host*

### Description

Specifies a demo license file or RepriseLM-based license used by DCV.
The value of the license_file parameter can be overridden by setting the environment variable DCV_LICENSE_FILE.

### Valid Values

The value for *license_file* can be either of the following:

- The full path name to the license file.

  Linux example:

  ```
  license_file = /opt/nice/dcv/license/license.lic
  ```

  Windows example:

  ```
  license_file = C:\Program Files\DCV\license\license.lic
  ```

- For a permanent license, the name of the license server host and TCP port number used by the rlm daemon, in the format *port@host*.

  For example:

  ```
  license_file = 5053@licserver
  ```

  The port number must be the same as that specified in the HOST line of the license file.

### Default Value

If you installed DCV with a default installation, the license file is expected to be in the license directory under DCV installation path.

## use_platform_product

### Syntax

use_platform_product = ...

### Description

Force the use of the old platform-specific products (dcv_linux, dcv_windows, dcv_windows_vm)

The value of the `use_platform_product` parameter can be overridden by setting the environment variable `DCV_LICENSE_USE_PLATFORM_PRODUCT`.

#### Valid Values

...

#### Default Value

`false`

## [Logging]

List of `[Logging]` settings.

### log_dir

#### Syntax

`log_dir` = *directory*

#### Description

Specifies the fullpath of the directory where DCV will create log files
The value of the `log_dir` parameter can be overridden by setting the environment variable `DCV_LOG_DIR`.

#### Valid Values

A path where the user running the DCV server can write.

### log_file

#### Syntax

`log_file` = *filepath*

#### Description

Specifies the full file path prefix used by DCV to create log file to which DCV messages are logged into. Each log file will be created in the form: *log_file.pid*, where *pid* is an integer identifying the OpenGL process using DCV.

For example:

```
log_file = /tmp/dcv.log
```

will make DCV create log files as:

```
/tmp/dcv.log.1234
```

The value of the `log_file` parameter can be overridden by setting the environment variable `DCV_LOG_FILE`.

#### Valid Values

A file path where the user running the DCV application can write.

## log_level

### Syntax

log_level = *string*

### Description

Specifies the logging level of DCV messages.

For example:

```
log_level = INFO
```

The value of the log_level parameter can be overridden by setting the environment variable DCV_LOG_LEVEL.

### Valid Values

The log levels in order from highest to lowest are:

- error
- warning
- info
- debug

The most important DCV messages are at the error or warning level. Messages at the info and debug level are only useful for debugging.

### Default Value

warning

## [Remotization]

List of [Remotization] settings.

## allow_udp

### Syntax

allow_udp = ...

### Description

Whether to allow to use UDP.
The value of the allow_udp parameter can be overridden by setting the environment variable DCV_ALLOW_UDP.

### Valid Values

true | false

### Default Value

false

## blacklist_vchannels

### Syntax

`blacklist_vchannels` = :*backends*

### Description

List of DCV Virtual Channel backends to blacklist. When a vchannel is blacklisted its backend is not started and end stations will not be able to use it.
The value of the `blacklist_vchannels` parameter can be overridden by setting the environment variable `DCV_BLACKLIST_VCHANNELS`.

### Valid Values

List of backends separated by ":" on Linux and ";" on Windows

## codec_qu

### Syntax

`codec_qu` = *string*

### Description

Specifies the codec DCV should use for static image quality enhancement.

See also the `enable_update_quality` setting for enabling the static image quality enhancement.
The value of the `codec_qu` parameter can be overridden by setting the environment variable `RVN_CODEC_QU`.

### Valid Values

Supported codecs:

- `jpeg`: lossy JPEG compression;
- `rle`: lossless run-length encoding.
- `zrle`: lossless run-length encoding.
- `lzorle`: lossless run-length encoding.

### Default Value

`lzorle`

## codec_tcp

### Syntax

`codec_tcp` = *string*

### Description

Specifies the codec DCV should use on TCP connections.
The value of the `codec_tcp` parameter can be overridden by setting the environment variable `RVN_CODEC_TCP`.

### Valid Values

Supported codecs:

- `h264`: lossy H264 compression;
- `jpeg`: lossy JPEG compression;
- `rle`: lossless run-length encoding.
- `zrle`: lossless run-length encoding.
- `lzo`: lossless run-length encoding.

### Default Value

`h264`

## default_proxy_port

### Syntax

`default_proxy_port` = *integer*

### Description

Specify the TCP port number of the HTTP or SOCKS5 proxy server that the DCV end station will use as a default to connect to the DCV server. The proxy properties on the end station will override these defaults.
The value of the `default_proxy_port` parameter can be overridden by setting the environment variable `DCV_DEFAULT_PROXY_PORT`.

### Valid Values

A valid TCP port number.

### Default Value

8123

## default_proxy_server

### Syntax

`default_proxy_server` = *hostname* | *IP address*

### Description

Specify the hostname or IP address of the HTTP or SOCKS5 proxy server that the DCV endstations will use as a default to connect to the DCV server. The proxy properties on the end station will override these defaults.
The value of the `default_proxy_server` parameter can be overridden by setting the environment variable `DCV_DEFAULT_PROXY_SERVER`.

### Valid Values

Hostname or IP address of a HTTP or SOCKS5 proxy server. Do not specify this setting if you want to directly connect to the DCV server without using any proxy server.

### Default Value

No proxy server.

**Note**

Proxy servers that require authentication are not supported. This may be the subject of a future release.

## `default_proxy_type`

### Syntax

`default_proxy_type` = `HTTP | SOCKS5`

### Description

Specify the type of proxy server that the DCV end station will use as a default to connect to the DCV server. The proxy properties on the end station will override these defaults.
The value of the `default_proxy_type` parameter can be overridden by setting the environment variable `DCV_DEFAULT_PROXY_TYPE`.

### Valid Values

`HTTP` to use a HTTP proxy or `SOCKS5` to use a SOCKS5 proxy

### Default Value

`HTTP`

## `enable_quality_update`

### Syntax

`enable_quality_update` = *boolean*

### Description

Specify if DCV must send a higher quality image (the *quality update*) when the 3D image is not changing.

See also the `quality_update_quality` setting for tweaking the quality level of the static images.
The value of the `enable_quality_update` parameter can be overridden by setting the environment variable `RVN_ENABLE_QUALITY_UPDATE`.

### Valid Values

true | false

### Default Value

`true`

## `encryption`

### Syntax

`encryption` = alwaysmaximum | alwayson | preferon | preferoff

### Description

The server application encryption mode.

The value of the `encryption` parameter can be overridden by setting the environment variable `DCV_ENCRYPTION`.

### Valid Values

`alwaysmaximum` to use always encryption with a 256 bits encryption key `alwayson` to use always encryption with a 128 bits encryption key `preferon` to use 128 bits encryption if the endstation allows the server to decide or if sets preferon `preferoff` to not use encryption

### Default Value

`preferoff`

## host

### Syntax

`host` = *host*

### Description

Specifies the hostname or IP address of the external rendering server that the OpenGL applications must use to delegate 3D rendering. In case it is specified it must be a valid address reachable from the end station.
The value of the `host` parameter can be overridden by setting the environment variable `RVN_RENDERING_HOST`.

### Valid Values

Hostname or IP address of a DCV rendering server.

## image_quality

### Syntax

`image_quality` = *integer*

### Description

Specify the quality level of dynamic images when using TCP connections. Higher values correspond to higher image quality and more data transfer. Lower values reduce quality and reduce bandwidth usage.
The value of the `image_quality` parameter can be overridden by setting the environment variable `RVN_IMAGE_QUALITY`.

### Valid Values

1-100

### Default Value

`80`

## quality_update_interval

### Syntax

`quality_update_interval` = *integer*

### Description

Interval in milliseconds before a quality update is sent after a dynamic quality frame was sent.

The value of the `quality_update_interval` parameter can be overridden by setting the environment variable `RVN_QUALITY_UPDATE_INTERVAL`.

### Valid Values

Values greater than zero are valid.

### Default Value

`300`

## quality_update_quality

### Syntax

`quality_update_quality` = *integer*

### Description

Specify the quality level of static images. Higher values correspond to higher image quality and more data transfer. Lower values reduce quality and reduce bandwidth usage. See also the `enable_update_quality` setting for enabling the static image quality enhancement.
The value of the `quality_update_quality` parameter can be overridden by setting the environment variable `RVN_QUALITY_UPDATE_QUALITY`.

### Valid Values

1-100

### Default Value

`95`

## tcp_max_port

### Syntax

`tcp_max_port` = *integer*

### Description

End of the range of TCP ports used by DCV to listen to end stations.
The value of the `tcp_max_port` parameter can be overridden by setting the environment variable `RVN_MINI_MAX_PORT`.

### Valid Values

A valid TCP port number greater than `tcp_start_port`.

### Default Value

`7399`

## tcp_start_port

### Syntax

`tcp_start_port` = *integer*

### Description

Start of the range of TCP ports used by DCV to listen to end stations.

The value of the `tcp_start_port` parameter can be overridden by setting the environment variable `RVN_MINI_START_PORT`.

#### Valid Values

A valid TCP port number.

#### Default Value

`7300`

### `udp_quality`

#### Syntax

`udp_quality` = *integer*

#### Description

Specify the quality level of dynamic images when using UDP connections. Higher values correspond to higher image quality and more data transfer. Lower values reduce quality and reduce bandwidth usage.
The value of the `udp_quality` parameter can be overridden by setting the environment variable `RVN_UDP_QUALITY`.

#### Valid Values

1-100

#### Default Value

`80`

### `udp_start_port`

#### Syntax

`udp_start_port` = *integer*

#### Description

Start of the range of UDP ports used by DCV to listen to end stations.
The value of the `udp_start_port` parameter can be overridden by setting the environment variable `RVN_UDP_START_PORT`.

#### Valid Values

A valid UDP port number.

#### Default Value

`7300`

### `use_nvidia_hw_encoder`

#### Syntax

`use_nvidia_hw_encoder` = *boolean*

#### Description

Specifies to use the NVIDIA HW Encoder.

The value of the `use_nvidia_hw_encoder` parameter can be overridden by setting the environment variable `DCV_USE_NVIDIA_HW_ENCODER`.

#### Valid Values

true | false

#### Default Value

`true`

## use_udp

#### Syntax

`use_udp` = *integer*

#### Description

Specify whether DCV should use TCP or UDP connections.
The value of the `use_udp` parameter can be overridden by setting the environment variable `RVN_USE_UDP`.

#### Valid Values

0 | 1 | 2

#### Default Value

`0`

# [RenderingServer]

List of `[RenderingServer]` settings.

## agent_max_port

#### Syntax

`agent_max_port` = *integer*

#### Description

Maximum value of the ports interval used by the rendering agents.
The value of the `agent_max_port` parameter can be overridden by setting the environment variable `SVN_SOCKETS_MAX_PORT`.

#### Valid Values

A TCP port number.

#### Default Value

`60000`

## agent_start_port

#### Syntax

`agent_start_port` = *integer*

#### Description

Start of the ports interval used by the rendering agents.
The value of the `agent_start_port` parameter can be overridden by setting the environment variable `SVN_SOCKETS_PORT`.

#### Valid Values

A TCP port number.

#### Default Value

`34567`

## `host`

#### Syntax

`host` = *IP address*

#### Description

Specifies the network interface to which the rendering server should bind.
The value of the `host` parameter can be overridden by setting the environment variable `DCV_RENDERING_SERVER_HOST`.

#### Valid Values

The rendering host IP address to which application host are connected via a high bandwidth/low latency network.

## `local_display_list`

#### Syntax

`local_display_list` = :*display*[.*screen*] ...

#### Description

List of local 3D accelerated X displays and screens used by DCV for OpenGL rendering.
The value of the `local_display_list` parameter can be overridden by setting the environment variable `DCV_LOCAL_DISPLAY_LIST`.

#### Valid Values

Space separated list of local 3D accelerated X displays.

#### Default Value

`:0.0`

> **Note**
> Access to the selected displays must be granted to the user running the rendering server.

## `port`

#### Syntax

`port` = *integer*

#### Description

Specifies the TCP port where the rendering server should listen on.
The value of the `port` parameter can be overridden by setting the environment variable `DCV_RENDERING_SERVER_PORT`.

#### Valid Values

A free TCP port number above 1024.

#### Default Value

`2007`

### user

#### Syntax

`user` = *operating system username*

#### Description

Specifies the user account the rendering server is running as.
The value of the `user` parameter can be overridden by setting the environment variable `DCV_RENDERING_SERVER_USER`.

#### Valid Values

Any unprivileged operating system username.

#### Default Value

`dcv`

## [Wrapper]

List of `[Wrapper]` settings.

### disable_driver_unload

#### Syntax

`disable_driver_unload` = *boolean*

#### Description

Specifies if explicit unload of the driver's GL library must be disabled.
The value of the `disable_driver_unload` parameter can be overridden by setting the environment variable `DCV_DISABLE_DRIVER_UNLOAD`.

#### Valid Values

true | false

#### Default Value

`true` - Unload GL library