cadence®

Virtuoso Design Environment User Guide

Product Version ICADVM20.1
October 2020

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

11
Using Bookmarks and Views 253

Preface

The Virtuoso® Design Environment User Guide provides an overview and introduction to the features and functionality that will be encountered while working with the Virtuoso platform and an understanding of the new operating environment in the IC6.x releases.

This user guide is aimed at developers and designers of integrated circuits and assumes that you are familiar with:

- The Virtuoso design environment and application infrastructure mechanisms designed to support consistent operations between all Cadence® tools.
- The applications used to design and develop integrated circuits in the Virtuoso design environment, notably, the Virtuoso Layout Suite, and Virtuoso Schematic Editor.
- The Virtuoso design environment technology file.
- Component description format (CDF), which lets you create and describe your own components for use with Layout XL.

This preface contains the following topics:

Scope

The functionality described in this guide can be used only in advanced nodes, such as ICADVM20.1.

Licensing Requirements

For information on licensing in the Virtuoso design environment, see the *Virtuoso Software Licensing and Configuration Guide*.

Related Documentation

What's New and KPNS

- *Virtuoso Design Environment What's New*
- *Virtuoso Design Environment Known Problems and Solutions*

Installation, Environment, and Infrastructure

- *Cadence Application Infrastructure User Guide*
- *Cadence Hierarchy Editor User Guide*
- *Cadence Library Manager User Guide*
- *Virtuoso Software Licensing and Configuration Guide*

Additional Learning Resources

Video Library

The Video Library on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see Virtuoso Videos.

Rapid Adoption Kits

Cadence provides a number of Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

In addition, Cadence offers the following training courses on the SKILL programming language:

- SKILL Language Programming Introduction
- SKILL Language Programming
- Advanced SKILL Language Programming

To explore the full range of training courses provided by Cadence in your region, visit Cadence Training or write to training_enroll@cadence.com.

Note: The links in this section open in a separate web browser window when clicked in Cadence Help.

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.
- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the

Home button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in *Virtuoso Design Environment User Guide.*

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.

- Log on to Cadence Online Support

  Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type exactly as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
|	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.
`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument. Used with brackets to indicate that you can specify zero or more arguments. Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence® SKILL® language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\ ) indicates that the current line continues on to the next line.

1

Introducing the Virtuoso Design Environment

The Virtuoso design environment features applications that have:

- Sophisticated design window management including
  - Tabbed window architecture (for simultaneous editing of several cells and views)
  - Dockable assistant panes (for access to property editing and design-related tasks)
- Improved hierarchical navigation
- Smart interactive search technology

See the following topics for more information:

- Understanding the Operating Environment
- Starting the Virtuoso Design Environment
- Getting Started with Virtuoso Software
- Using the Command Interpreter Window
- Working with Cellviews
- Getting Started with Workspaces
  (including Sample Session Window and Workspace)
- Using Bookmarks and Views
- Navigating Cellviews and Hierarchies

See also Glossary of Terms.

For more advanced information, see the following topics:

- Customizing Your Design Environment on page 125
- Specifying Environment Settings on page 173

■ Bindkeys and Access Keys on page 309

Understanding the Operating Environment

A *window manager*—the Common Desktop Environment (CDE), for example—controls the size, placement, and behavior of windows. You can adjust various settings to customize your windowing environment:

- You can customize settings that affect only the Cadence software using menu selections in the software or by editing the `.cdsinit` and `.cdsenv` files.

  For more information, see Customizing Your Design Environment

- You can customize X Window System settings—which affect the look, feel, and behavior of both Cadence windows and other windows on your screen—in your `.Xdefaults`, `.xsession`, and `.xinitrc` files.

  For information on high-level system setups, see the *Virtuoso Software Licensing and Configuration Guide*.

Starting the Virtuoso Design Environment

Virtuoso Design Environment software is built into binary files—or *workbenches*—containing blocks of executable computer code. Each workbench contains the code to run several related applications.

Note: From IC614, `virtuoso` is the recommended workbench. The `layout` (and `layoutPlus`) workbenches are being phased out from IC614, but will still be accessible via symlinks in the short-to-medium term. It is however recommend that you now use the `virtuoso` executable.

When you start the Virtuoso Design Environment, the Command Interpreter Window (CIW) opens (see Chapter 4, "Using the Command Interpreter Window"). From the CIW, you can access Cadence applications that you are licensed to run.

To start a workbench, type the workbench binary name at the system prompt. For example, to start the `virtuoso` workbench, type `virtuoso` at the system prompt. Your system administrator can tell you the command to type to run your particular set of applications.

Caution

The installation procedure puts your Cadence executables in *your_install_dir*`/tools/dfII/bin`***. You must not move any of the Cadence executables from this location or they will not run.***

You can run Virtuoso Design Environment software in graphics or nongraphics mode:

- When you start the software in graphics mode, the Command Interpreter Window (CIW) appears (see Chapter 4, "Using the Command Interpreter Window"). From the CIW, you can start individual Cadence applications.
- In nongraphics mode, you can type Cadence SKILL commands or do any other work that does not require graphic display of designs or the graphical user interface. You can start Cadence software in nongraphics mode using the `-nograph` command-line option (see "Starting Cadence Software" on page 36).

When Virtuoso has launched, the time taken to successfully checkout the 111 (Virtuoso Framework) license will be shown in the CIW, for example:

```
\o Virtuoso Framework License (111) was checked out successfully. Total checkout
time was 0.32s.
```

You can now go on to access some applications by selecting a menu item (such as from the *Tools* menu in the CIW) while other applications start automatically when you open a design cellview (using *File – Open*). For example, when you open a schematic, your schematic editor starts automatically.

Key Terms

Term	Definition
Active window	The window that receives input. This is usually the window where the mouse pointer is located. The border of this window has a different color than those of the other windows.
Binary	A block of executable computer code containing the code that runs a Cadence application.
Click*	Press and release the left mouse button.
Command Interpreter Window (CIW)	The window that controls the Cadence design session. You use the CIW to type commands, start applications, and open other Cadence windows. For more information see Chapter 4, “Using the Command Interpreter Window,”.
Current window	In Cadence software, the window where an action takes place. This might not be the same as the active window. For example, you type commands in the Command Interpreter Window (active window) that have an effect in the design window (current window).
Design window	The window that displays your design.
Drag*	Press and hold the mouse button and move the mouse.
Double-click*	Click twice, rapidly.
Icon	A small graphic representation of a window, file, application, command or state
Press and hold*	Press the mouse button and do not release it.
Root window	The entire screen, which serves as a background for the other windows.
`Shift`-click*	Press and release the mouse button while holding down the `Shift` key.
Terminal window	A window where you can type UNIX commands.

*When these terms are used without specifying a button, use the left button.

Interacting with the User Interface

The following topics are discussed:

Using the Pointer and the Cursor

- The pointer is the visible shape that moves across the screen as you move the mouse.
- The cursor is a visible shape (see table below and refer to your own desktop documentation) in each window that shows where you can enter text.

  The cursor can have one of several shapes—typically a rectangle in a terminal window or an I-beam in a Cadence form.

The pointer can have many shapes, as shown in the examples below:

Example Shape	When you see it
	When the pointer is in the title bar (window environment dependent) of a window you are moving.
	When you resize the window using edges. You can increase or decrease the width of the window by moving one side of it.
	When you resize the window using corners. You can enlarge or reduce the window by moving two sides (the corner) of the window.
	When you are pointing in a window. You can select commands or objects.

Instructions in this user guide assume that the window with the pointer in it is the active window.

To verify that the pointer determines the active window, move the pointer to another terminal window and type some text. If the text appears in the window containing the pointer, the pointer determines the active window.

Using Pop-Up Menus and Forms

Some menus and forms are invisible during ordinary use but can be displayed (popped up) when you need them. Use the following mouse actions to pop them up:

To	Do this
Pop up the root menu	Click or press and hold the middle button in the root.
Pop up the window manager menu from an icon	Click or press and hold the right mouse button over the icon. **Note:** This is desktop and/or window manager dependent.
Pop up a Virtuoso menu	Click the right button in a graphics window.
Pop up an options form (if one exists)	Press the F3 key.
Choose an item on any pop-up menu	If holding down a mouse button, slide the pointer to the item and release, otherwise click the item.

Using Cadence Menus

A menu is a list of related commands. Cadence software uses the following kinds of menus:

- Pull-down menus: accessed from the menu banner of a window
- Pop-up menus: accessed with the mouse while you work in a design window
- Toolbar icons: found on main window and assistant toolbars
- A floating icon, typically a column or large icon buttons.

When you use an icon on the toolbar, the window automatically becomes active and the command affects only that window. Inactive buttons are grayed out and do not respond when you click them (they will also not be highlighted when under mouse cursor focus).

Selecting and Deselecting Items

To select and deselect items, do the following:

To	Do this
Select a single item	Click the item.
Select several consecutive items	Move the pointer to the first item, press and hold the left mouse button, and drag the pointer until all the items you want are highlighted.
Extend a selection	`Shift`-click on the new end point.
Delete one item from a selection	Press the `Control` key and click the item.
Delete several items from a selection	`Control`-click and drag the pointer over the items to delete. When you are done, click *OK*.
Close the box without selecting an item	Click *Cancel*.

The Cadence SKILL Language

When you enter commands from a form or menu, the system calls functions written in the SKILL programming language. Cadence software uses the Cadence SKILL language (modeled after the artificial intelligence language LISP) to communicate within and between applications and designs.

SKILL is available to Cadence users. You can enter SKILL commands from the input line of the CIW or place them in command files. You can use SKILL for small tasks, such as issuing a single command, or you can build complex routines to customize your system and add functionality.

The SKILL development tools is a set of software applications to help you write and debug SKILL code. To access these applications, choose *Tools – SKILL IDE* from the CIW. For details, see *Cadence SKILL IDE User Guide*.

Finding Information about the SKILL Programming Language

For information about the SKILL programming language, see

- *Cadence SKILL Language User Guide*
- *Cadence SKILL Language Reference*
- *Virtuoso Design Environment SKILL Reference*
- *Cadence User Interface SKILL Reference*
- *Cadence SKILL IDE User Guide*
- SKILL manuals for individual application programs

Using SKILL Finder to Find Information about SKILL Functions

For quick reference information about SKILL functions, use the SKILL Finder, a software utility that gives the syntax, description, and values returned for all the available SKILL functions.

To access the SKILL Finder, do the following:

➡ From the CIW, choose *Tools – SKILL Finder*.

The Cadence SKILL API Finder window opens.

File Storage

Cadence applications use the `./.cadence` directory to store various files including the recently-visited file lists, bookmark files, and preferences for workspaces, toolbars, and assistant panes. For more information on the `.cadence` hierarchy, see Understanding the .cadence Hierarchy.

The program stores design navigation information, which you can access from the Go toolbar, in `./.cadence/dfII/history/`*`username`*`.history`.

Important

The Cadence software creates and edits the files in the `.cadence` hierarchy. You should avoid direct editing of the contents of `.cadence`. Cadence will also install a signature file too to verify that Cadence has created this directory.

You can choose to save your custom workspace and toolbar files (see Customizing a Cadence Workspace and Customizing Toolbars) to another directory; for example, *`selected_path`*`/dfII/workspaces/`*`application`*. You can use the `setup.loc` file to specify your search path preferences.

A typical `setup.loc` might contain the following:

`.`	The current directory.
`$CDS_WORKAREA`	The user work area, if defined
`$CDS_SEARCHDIR`	The search directory, set by some tools
`startup`	The startup location
`$HOME`	The home directory
`$CDS_PROJECT`	The project storage area
`$CDS_SITE`	The site setup information
`$(compute:THIS_TOOL_INST_ROOT)/share`	The Cadence default setup information

For more information about the `setup.loc` file, see the Cadence Setup Search File: setup.loc in *Cadence Application Infrastructure User Guide*.

ICADVM20.1 Hardware and Software Requirements

This section provides hardware, software, and file system requirements for the ICADVM20.1 release. The workstation on which you install ICADVM20.1 and all client workstations must meet these requirements.

Two sets of requirements are provided: minimum and recommended. Meeting the *minimum* requirements lets you:

- Install and license your product
- Run product tutorials or other sample designs
- Create and execute a medium-size design
- Secure hotline support

In addition, meeting the *recommended* requirements allows you to do everything possible with the minimum requirements, plus:

- Create and execute a large design with the operating performance stated in product literature.

System Configuration Checking Tool

The System Configuration Checking Tool is available for Linux platforms. Use it to verify that a workstation on which you want to install the product meets the minimum hardware and software requirements, including operating system version, memory, swap space, and patch requirements.

To use the System Configuration Checking Tool, go to

System Configuration Checker (checkSysConf).

Note: This tool only checks that your workstation meets a subset of the requirements to run the product. Even if you use this tool to check these basic requirements, you must still make sure your workstation meets all requirements listed in this document.

Important

The following minimum requirements vary from design to design and are intended to be guidelines only. Simple designs will be less demanding on hardware than complex designs. Also, highly complex designs can overtax the minimum requirements described in this document. Designs of this type might work best on a high-end configuration.

Graphic Card Requirements for Desktop Workstations

For screen resolutions greater than 1600x1200 (including multiple monitor setups), the minimum size of graphics memory should be at least 64Mbytes per 1600x1200 screen. If this requirement is not met, the maximum number of Virtuoso windows in use, and their performance, will be significantly reduced. For higher resolutions, the Video Memory should be increased.

Also, the Xserver should not have any artificial memory limit set if the Xserver is installed on the client side of a client-server configuration. An artificial memory limit can reduce performance and window capacity.

On some thin client platforms, such as the Citrix, the Xredner extension of Xserver may face some issues while displaying halos and transparent shapes correctly. You can use the following shell environment variable to disable this Xrender extension.

```
setenv CDS_DISABLE_XRENDER_GRAPHICS true
```

XResources Requirements

If the initial allocated X server resource count is less than 2 million, significant performance impact may be felt when the X server needs to recycle these resources. Smaller values, such as 1M or 512K can cause display issues.

Note: X resource information is recorded in the CDS.log header.

Network Requirements

Good network performance is crucial to a good user experience with Virtuoso. This is particularly important when a user is remotely displaying Virtuoso from a server farm machine onto a local desktop.

There are two major aspects to network performance:

- High network bandwidth is important to sustain high data volume operations such as “redraw” of a chip level layout.
- Low network latency is important for good interactive experience and avoiding issues such as lagging mouse cursor. We can tolerate up to 150ms of latency, but we recommend it to be less.

Linux

Hardware Requirements for Linux Desktop		
	Minimum	**Recommended**
Graphics Card	AGP graphics card with at least 128 MB video memory	PCI-X graphics card with at least 256 MB video memory
Display	24-bit color display	24-bit color display
Physical Memory	64-bit: 16 GB	64-bit: 32 GB
Disk Space	64-bit: 100 GB	Dependent on design complexity

Hardware Requirements for Linux Server		
	Minimum	**Recommended**
Display	24-bit color display	24-bit color display
Physical Memory per active user	64-bit: 16 GB	64-bit: 32 GB
CPU per active user	1 core	2 cores
Disk Space	64-bit: 100 GB	Dependent on design complexity

Software Requirements for Linux	
Operating System	Red Hat Enterprise Linux (RHEL) 6.5 or later; or SUSE Linux Enterprise Server (SLES) 11 or later
Window Manager	GNOME/KDE
Compiler	gcc 6.3.0, g++ 6.3.0
Java Version	8

Patch Recommendations for Linux

For ICADVM20.1, Cadence recommends that you update your workstations to the platform patches listed at:

Recommended platform patches for systems running Cadence products

Alternatively, to go through Cadence Online Support (COS) follow these steps:

1. Log on to Cadence Online Support.
2. On the home page, click *Software – Computing Platforms*.
   The Computing Platforms web page is displayed.
3. In the Computing Platforms section, click *Recommended patches for systems running Cadence products*.

Also run the `checkSysConf` script to verify that you have all the patches required for this release.

2

Getting Started with Virtuoso Software

Before getting started with Virtuoso software, you should read Chapter 1, “Introducing the Virtuoso Design Environment,” which provides links to topics that can assist you in working in the Virtuoso Design Environment.

In this chapter, you can read about the following topics:

See also “Getting Help” on page 81.

Starting Cadence Software

Note: For information about setups that take place before you start the software, see the *Virtuoso Software Licensing and Configuration Guide*.

To start Cadence software from a UNIX prompt, do the following:

1. Change to the directory where you start the Cadence software:

   `cd` *`yourDirectory`*

   In Cadence software, this directory becomes your current directory.

2. Set environment variables as necessary (see Setting UNIX Environment Variables in the *Cadence Library Manager User Guide* for more information).

3. Type the command to start the Cadence software.
   (You can ask your system administrator for the correct command.)

When Cadence software starts, it reads the `.cdsinit` file to set up your Cadence environment. Then the software searches the following three locations in the order listed and uses the first `.cdsinit` file it finds:

1. *`your_install_dir`*`/tools/dfII/local`
2. The directory from which you started the Cadence software (`.`)
3. Your home directory (`~`)

Note: CSF can be used for the `.cdsinit` file.

The software reads no further unless this `.cdsinit` file instructs the software otherwise.

You can customize your work environment by editing system and application setup files.

See also

- Command-Line Options on page 37
- Getting Started Examples on page 48

Command-Line Options

You can use the following command-line options when starting any Cadence application:

`-45`	Provides enhanced drawing of 45-degree diagonal lines. With this option, diagonal lines are not jagged. This behavior might vary from shell to shell.
`-help`	Lists command-line options.
`-log fileName`	Specifies the log file (*`fileName`*) to record this session. For example, use the following command to save the log file of the `virtuoso` executable in the `myLogDir` directory using a name that reflects the date, such as `CDS23Jul16:35.log`: `virtuoso -log ~/myLogDir/CDS"date '+%d%h%H%M'".log`

`-logtime`

Adds an absolute timestamp for every entry in the log file.

By default, the absolute time is logged in UTC (Coordinated Universal Time or GMT) to allow timestamps that can be compared between timezones. The first two lines of the log file display the start time in UTC as well as local time.

Time can be logged as the local time rather than UTC by setting the environment variable `CDS_USE_LOCAL_TIMESTAMP` to true:

```
setenv CDS_USE_LOCAL_TIMESTAMP True
```

This environment variable also controls whether other timestamps in the `CDS.log` file, such as the timestamp for memory reports, use the local time instead of UTC. When this environment variable is used, the first line of the log file displays the local start time along with the UTC offset.

Timestamp logging can also be enabled by setting the environment variable `CDS_LOG_TIMESTAMPS` to true:

```
setenv CDS_LOG_TIMESTAMPS True
```

Timestamps are output to the nearest millisecond, but the resolution of the timestamps is determined by the hardware on the system on which the application is run.

Log files with timestamps can be replayed but the timestamps are ignored during replay. Log files with timestamps are not replayable in the older releases that did not have the timestamp feature.

Example:

2008.01.30 21:02:07.357\i load "allFields.il"

2008.01.30 21:02:07.512\t t

2008.01.30 21:02:07.514\p >

2008.01.30 21:02:12.060\i ipcSleep(5)

2008.01.30 21:02:17.104\t t

2008.01.30 21:02:17.114\p >

2008.01.30 21:02:22.523\i hiDisplayForm phzForm2D

2008.01.30 21:02:26.110\o "Mapping form: phzForm2D"

2008.01.30 21:02:27.174\t t

2008.01.30 21:02:27.176\p >

2008.01.30 21:02:28.899\i ipcSleep(5)

2008.01.30 21:02:33.904\t t

2008.01.30 21:02:33.913\p >

2008.01.30 21:02:35.930\i exit

2008.01.30 21:02:35.931\o

`-logtimerel`

Adds a timestamp relative to the previous log entry in the log file. This allows you to see the amount of time that elapsed between entries.

Also, the first two lines of the log file display the start time in UTC (Universal Time or GMT) as well as local time.

Timestamps are output to the nearest millisecond, but the resolution of the timestamps is determined by the hardware on the system on which the application is run.

Log files with timestamps can be replayed but the timestamps are ignored during replay. Log files with timestamps are not replayable in older releases that did not have the timestamp feature.

00:00:00.030\i load "allFields.il"

00:00:00.105\t t

00:00:00.012\p >

00:00:00.006\i ipcSleep(5)

00:00:05.012\t t

00:00:00.009\p >

00:00:00.003\i hiDisplayForm phzForm2D

00:00:02.826\o "Mapping form: phzForm2D"

00:00:00.583\t t

00:00:00.011\p >

00:00:00.007\i ipcSleep(5)

00:00:05.012\t t

00:00:00.009\p >

00:00:00.005\i exit

00:00:00.001\o

`-maxload`	Specifies the maximum number of CPU-bound tasks that can be executed simultaneously. The default value for `maxload` is the number of CPU cores in a machine. The `maxload` value should not exceed the maximum number of threads allowed in a Virtuoso session, which is either defined by `setrlimit` or by the `-maxthreads` option. A task is not 100% CPU-bound if it takes less that 1 CPU core load.
`-maxthreads`	Specifies the maximum number of threads that can be created in a Virtuoso session. The default value for `maxthreads` is driven by the hard limit set by `setrlimit`. If the number of threads in a session exceeds the `maxthreads` value set by the hard limit, the application exits and a warning message is displayed.
`-noblink`	Disables blinking. When blinking is disabled, objects that would normally blink in graphics windows will not blink. In many cases, disabling blinking can improve graphics performance, particularly for many VNC X Window setups. You can also use the SKILL function `hiEnableBlink` to enable or disable blinking while you are running the application and `hiIsBlinkEnabled` to check if blinking is enabled.
`-nocdsinit`	Does not read any `.cdsinit` files.

`-nograph`

Specifies non-graphics mode, which you can use for SKILL programming and replaying log files.

In this mode, Unix Domain Socket (UDS) is used to connect to the nograph server. The TCP port is not opened.

This mode uses the Cadence-provided VNC server, `cdsXndx`. By default, Cadence uses display numbers between :90 and :95 inclusive to communicate with the non-graphical server. Therefore, these numbers should not be used for X Window displays on hosts that run Cadence software using `-nograph`. The program will also write log files into the directory `$HOME/.vnc-cds`, therefore `$HOME` must have write permissions when `-nograph` is being used.

Use this option carefully as it is intended primarily only for replaying log files (which were generated in graphics mode) for testing purposes.

`cdsXvnc` can leave locked files in `/tmp` if it exits abnormally (for example, it is killed or crashes). In such cases you may be displayed warning messages like:

```
*WARNING* Could not open display sfsol804:80.
If no X server is running for
*WARNING* this display, remove the file /tmp/
.X11-unix/X80.
*WARNING* Trying another display.
*WARNING* Could not open display sfsol804:81.
If no X server is running for
*WARNING* this display, remove the file /tmp/
.X11-unix/X81.
*WARNING* Trying another display.
```

when starting in `-nograph` mode. If no VNC session or other X server is running and using that display, then the lock file should be removed if this message is output.

See also: Using XVNC.

`-nographE`

Emulates the non-graphical mode using an X server other than `cdsXndx` (the Cadence-supplied non-graphical X server). This option is similar to `-nograph` except that it uses the default display or the display specified with either the `DISPLAY` environment variable or the `-display` command line option. Windows and forms will be drawn on the specified display.

You can also start the non-graphical emulation mode by setting the following environment variable:

CDS_NOGRAPH_DISPLAY *`display`*

and using either the `-nograph` or `-nographE` option to start the application.

If you use both `-display` and `CDS_NOGRAPH_DISPLAY`, then the `-display` value overrides that of `CDS_NOGRAPH_DISPLAY`.

`-nographN`

Starts the non-graphical mode using the display specified by the argument N, where N can be a number 0 through 9 or "+". The "+" option displays the application in the :400 through :409 display range.

Do not use the `-nograph+` option with any of the tenbase options. Use the `-nographN` option instead.

`-nologtime`

Overrides the setting made by the environment variable `CDS_LOG_TIMESTAMPS` and disables timestamp logging.

`-noxshm`

Prevents use of X Shared Memory Access (XSHM), which is a protocol used for communicating with the X display server to enable better performance. This mode can only be used if the X server is running on the same machine as the Virtuoso software.

Use the `-noxshm` option in the following situations:

- If you are using Virtuoso (graphically) with an X display configuration that appears to be local but is in fact remote and if the X server supports the XSHM extension.
- If you see BadAccess and BadShmSeg error messages when using Virtuoso graphically

Note: Starting from this release, if shared memory access fails to work, Virtuoso automatically stops using it.

`-replay fileName`

Executes a SKILL file (*`fileName`*).

Note: This option allows replaying a log file from a previous virtuoso session. The font and disclosure settings are not read at startup during replay so these are always started with default settings.

`-restore fileName`

Restores a saved session from *`fileName`*.

Note: This option cannot read a log file because it uses the SKILL function `load` to load the file toward the end of the initialization process.

`-showCtrlFile`

Displays the control file content. All Virtuoso early access releases have a limited lifespan to ensure that customers do not use these short-term releases forever. Therefore, a control file is given to the customers that contains an expiration date and the sub-version value of the release.

To view the control file content, you need to run the following command:

```
virtuoso -showCtrlFile
```

Note: After you use this option, the Virtuoso session is terminated.

`-V`

Returns the release number, such as 4.4, without starting the Cadence software.

`-W`	Returns the version number without starting the Cadence software. To identify the exact software version number, type a command similar to the following in a terminal window. *`your_install_dir`*`/tools/dfII/bin/virtuoso -W`

See the user guide for your application for other options that are specific to that application.

Note: The software uses a 24-bit TrueColor visual. If it cannot find that, it looks for a 16-bit TrueColor visual. If it cannot find that either, it looks for a 15-bit TrueColor visual. If it cannot find any of these, it will not run.

 *Tip*

To get information about your visual or visuals and the color depth of each visual, do the following:

- For detailed information about your visuals, type `xdpyinfo` at the system prompt.
- For root visual information, type `xdpyinfo | grep root` at the system prompt.

The following section describes using XVNC, which is a Unix VNC server based on a standard X server, to work on Virtuoso.

Using XVNC

For Virtuoso, `XVNC` is controlled through the following environment variables and command line options:

The environment variables are:

- `CDS_XVNC_OFFSET` (which can be a number from 0-9)
- `CDS_XVNC_TENBASE` (which can be a number from 1-9 or "+")

The command line options are:

- `-nographN` (where `N` can be any digit from 0-9)
- `-vncTenbase N` (where `N` can be any digit from 1-9 or "+")

Note: `-nographN` is one word, whereas `-vncTenbase N` are two.

If `-nographN` is specified, it overridesthe `CDS_XVNC_OFFSET` settings. If -`vncTenbase N` is specified, it overridesthe `CDS_XVNC_TENBASE` settings.

CDS_XVNC_OFFSET are the recommended settings, however, if it is not defined, the CDS_XNDX_OFFSET settings work.

Note: -nographN overrides both CDS_XVNC_OFFSET and CDS_XNDX_OFFSET settings.

By default, non-graphical mode (-nograph) is opened using the non-visible display :95. If display :95 fails, checks are performed to see if a lock file for this display exists.

If the lock file exists, after waiting for few seconds an attempt to open display :95 is made again. If this attempt also fails, a message indicating that the lock file should be removed is displayed (if no Xserver is running for that display) and an attempt to connect with the next display (:90 through :99 or from :99 to :90 if starting at a display other than :90) is made.

If no lock file exists, Virtuoso will use the cdsVncserver script to start the nograph server cdsXvnc for display :95. An attempt to connect to display :95 is made every 3 seconds until either the connection is made to this display, or 10 attempts are made (this typically takes about a minute, since the connection attempt itself can take a few seconds). If the connection fails in all of the ten tries, a failure message is displayed and an attempt to connect with the next display is made. This process is repeated until either a connection is made, or it has failed for all the displays :90 through :99.

- CDS_XVNC_OFFSET, or -nographN, will change the first display it tries to the one with the lowest digit taken from the environment variable or the option. So, for example, if the offset is set to 3, then the attempt to open display :93 is made first, then :94, and so on. When it reaches :99, if that fails, an attempt to open display :90 is made, and the last one is :92. Therefore, attempts are made to always try up to 10 displays, with the same first digit and starting with the second digit as 0 or that specified by the offset option.

- CDS_VNC_TENBASE, or -vncTenbase N, will change the first digit of the display. It is,however, limited to 1-9. For example, if -vncTenbase is set to 5, then the displays used are :50 through :59, starting at :55, unless the offset is specified (in which case it starts at :5N, where N is the offset). Then it works like the default range of :90 through :99, except in this case it will be :50 through :59.

Note: When VNC is used interactively, and a specific display is not specified, VNC by default starts with display :0, and then sequentially adds 1 until it finds an available slot. Normally, a VNC session will not be shared among multiple users. However, the cdsXvnc sessions are configured to allow sharing. So, multiple nograph users will share the same session, and multiple nograph sessions running on the same machine that use the same display number will share the same cdsXvnc server (assuming the number of sessions on that cdsXvnc server has not maxed out).

In fact, because multiple nograph sessions share the same cdsXvnc server, it is important not to kill a running cdsXvnc process. cdsXvnc exits automatically after all programs

connecting to it have disconnected. Killing a running `cdsXvnc` process will likely cause one or more Virtuoso sessions to fail and display an error.

Getting Started Examples

Before you start Virtuoso, do the following:

- Obtain the required licenses and software. See, *Virtuoso Software Licensing and Configuration Guide*.
- Set up Cadence tools' executable binary paths for Virtuoso and other tools, such as MMSIM simulators. You may add these paths into your `.cshrc` file.
- If required, customize your virtuoso with the Virtuoso initialization file (`.cdsinit`) and the environment variable file (`.cdsenv`). See Specifying Environment Settings.

Note: Both `.cdsenv` and `.cdsinit` is read by Virtuoso in a designated order. For more information, see Specifying Environment Settings.

- To start Virtuoso from a UNIX prompt, do the following:

  a. Change to the working directory from where you want to start the software:
  `cd <your_working_directory>`

  b. Confirm if the Virtuoso executable is set properly:
  `which virtuoso`

  c. Define the design library and reference libraries (`cds.lib`).

  d. Type the command to start the Virtuoso software:
  `Virtuoso &`

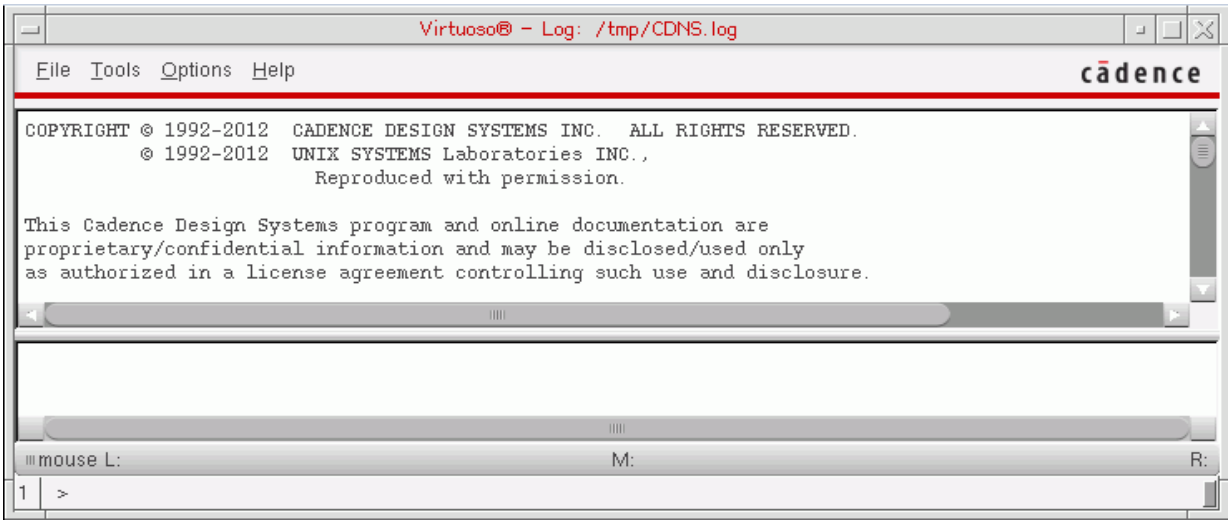
- Now, you can create and access your design project cellviews. If required, you can also set up the design data version control and management using either Cadence's Generic Design Management (GDM) (see the Overview section of the *Cadence Application Infrastructure User Guide*), or any other third party DM tool.

Examples of starting a Cadence workbench follow:

- To start Virtuoso as a background process, type the following command:

```
virtuoso &
```

The CIW appears.



➤ To start Virtuoso as a background process, with outputs being sent to a log file, type the following command:

```
virtuoso -log 1.log &
```

In this case, outputs sent to the terminal are also written to the log file, which can be accessed later, even if Virtuoso exits unexpectedly.

Alternatively, you can use the following options in Linux:

- Before closing the terminal, press `Ctrl+Z` to interrupt the process and use the `bg` command to put the process in the background.
- Launch Virtuoso using the `screen` command as follows:

  ```
  /usr/bin/screen virtuoso -log myLog
  ```

  In this case, even when the terminal running `screen` is closed, Virtuoso continues to run.

➡ To start Cadence software in a non-graphics mode, type the following command:

```
virtuoso -nograph
```

The CIW does not appear. The Design Environment prompt (>) appears.

Do not use an ampersand (&) after the command in nongraphics mode because you want to interact with the software rather than run the software in the background.

➤ To execute a SKILL file automatically on starting, type the following command:

```
virtuoso -nograph -replay fileName
```

where *fileName* is the name of the SKILL file you want to execute.

Saving Changes

You can save changes to design work, session defaults, window and forms position and size:

To save design work, do the following:

➤ From your application, choose *File – Save* or *File – Save As*.

To save changes to session defaults, do the following:

➤ From the CIW, choose *Options – Save Defaults*.

The Save Defaults form appears.

See Specifying New Default Values for more information.

To save the current position of windows and forms, do the following:

➤ From the CIW, choose *Options – Save Session*.

The Save Session form appears.

See Saving Window Positions for more information.
See also Chapter 9, “Getting Started with Workspaces” for information about saving positions of docked and floating assistant panes in your session window.

Tip

For ideas about how to preserve your session log file, see Saving and Renaming Log Files.

Saving and Renaming Log Files

The software writes a record of each session in the `CDS.log` file (unless you specify a different log file name using the `-log` option on the command line when you start your session). You can manage your session log files using any of the following methods.

- After you complete a session, you can rename the log file using the UNIX `mv` command. For example:

  ```
  mv CDS.log log.jun1
  ```

- When you start your session, you can use the `-log` option to specify a name for the log file. For example:

  ```
  startupCmd -log ~/log.jun1 &
  ```

  where *`startupCmd`* is the command you type to start your session (such as `virtuoso`).

- Before you start your session, you can set the `CDS_LOG_VERSION` environment variable to specify a unique name every time the software writes a log file (see Specifying the CDS.log File Name" in the *Cadence Library Manager User Guide*).

Quitting the Cadence Software

Important

Before you quit the Cadence software, you should save any design or data changes you made during this session (see "Saving Changes" on page 51).

You can exit the Cadence software by typing a command or selecting a menu item. The Design Environment prompt is available in both graphics and nongraphics modes. In graphics mode, a command prompt is available on the input line of the CIW (see also "Working with SKILL Commands" on page 85).

To quit the Cadence software from the Design Environment prompt, do the following:

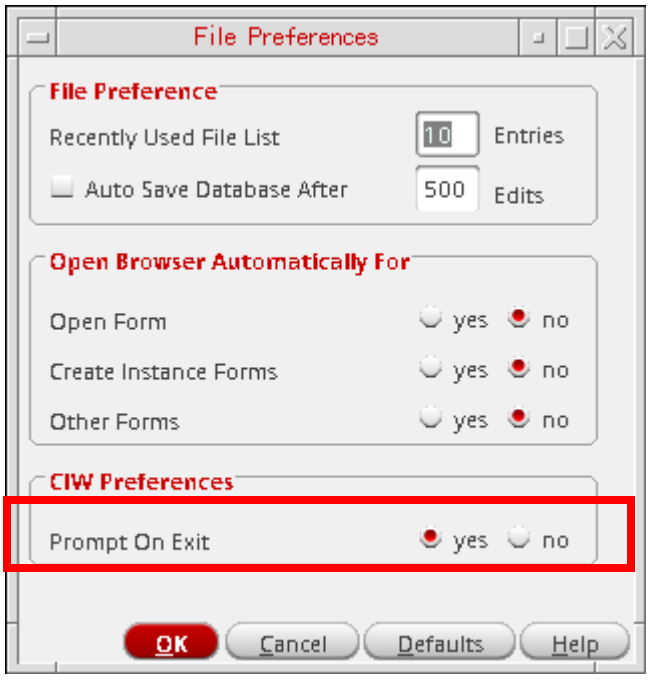
➤ Either on the CIW input line or at the nongraphics-mode Design Environment prompt, type `exit`.

To quit Cadence software from a menu, do the following:

1. From the CIW, choose *File – Exit*.

   An Exit prompt appears.

   Note: If you do not want an Exit prompt to appear every time you exit the Design Environment, you can select *no* for the *Prompt On Exit* option in the *CIW Preferences* group box on the File Preferences form (see "Specifying When the Exit Prompt Appears" on page 145).



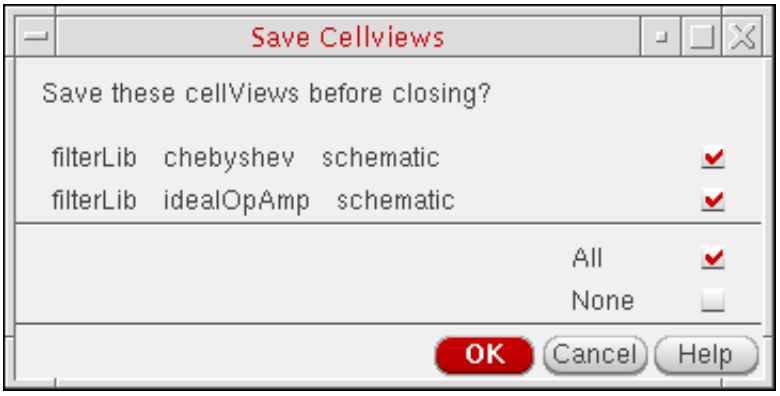
   You can also specify this preference by saving the following line in your `.cdsenv` file:

```
ddserv.ciw promptOnExit boolean nil
```

2. Click *Yes*.

   If you have made no changes since the last save, the software exits.

If you have unsaved data in any open session window, the Save Cellviews form appears.



A mark appears in the check box indicating those unsaved cellviews that the program will save when you click *OK*.

a. (Optional) Remove the mark from the check box for each cellview whose changes you do not want to save, or click *None* to deselect all cellviews and save none of your changes.

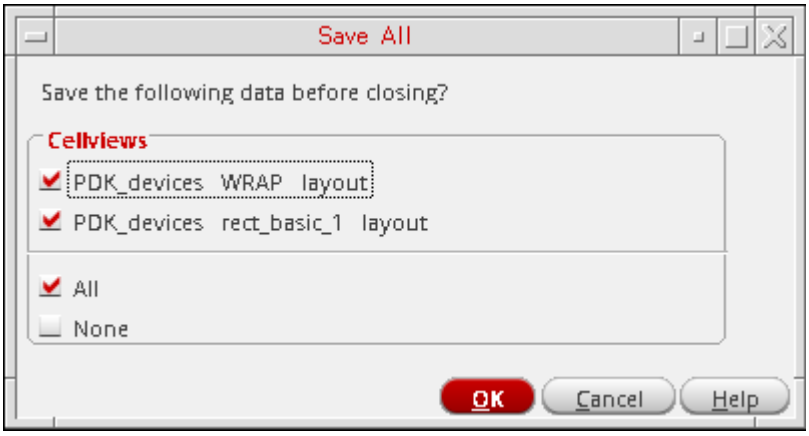
b. Click *OK*.

The specified changes are saved and the application exits.

Note: If you click *Cancel*, you cancel both save and exit operations. The program does not save your changes and does not exit.

Important

If you attempt to exit Virtuoso when there are more than 1632 modified cellviews, constraints, techfiles, and so on, active in your session, you will be prompted by the Save All form to choose to save All or None of these views and files; without the list of modified data being displayed as it exceeds displayable limits.



Backing Up Your Work

To back up your work, do the following:

1. Change to the directory you want to back up. For example:

   ```
   cd /usr/mnt/designs
   ```

2. Use the `tar` command to back up your work either to a tape drive or to another directory as follows:

   - To back up the `dfLib` library to a tape drive (*`tape_device`*), type the following:

     ```
     tar -cvf /dev/tape_device dfLib
     ```

   - To back up the `dfLib` library to another directory, type the following:

     ```
     tar -cvf - dfLib | (cd /usr/mnt/backup; tar -xf -)
     ```

We recommend that you back up your work regularly.

Unlocking an Application

The use of locks in a multi-user environment is an essential behavior as we do not want multiple edits taking place at the same time. The `.cdslck` file is therefore vitally important for ensuring this.

However, in certain circumstances, such as a crash, you may be required to remove an application lock.

The best method of doing this is to use the `clsAdminTool`, for example:

1. At a command line enter:

   ```
   % clsAdminTool
   ```

2. At the prompt enter either:

   `> -are .` (to remove all locks from the current directory)

   or

   `> -are <directoryLocation>` (to remove all locks from the specified location)

   Note: Enter "`help`" at the prompt to get a full list of the `clsAdminTool` options.

   You will then be provided with a list of the locks that have been removed, for example:

   ```
   BEGIN: Release Edit Locks.
    /user/jsmith/CDS.log lrd880a 13660 1173285728
    /user/jsmith/aspec/run/techManager.log kudos 10225 991131648
   ```

3

Obtaining Virtuoso Product Licenses

When you purchase a Virtuoso product, you also purchase licenses that entitle you to use the applications that are part of that product. Most of these licenses are *floating licenses* that are available to anyone in the work group. They are checked out from a central license pool at the beginning of work and checked back in when work is done. A few licenses might be restricted to specific users or servers.

When you run Virtuoso software, you typically use applications in combination with each other and the software checks out the necessary licenses automatically when you run the command to start an application. Alternatively, you can check licenses in and out from the Command Interpreter Window (CIW).

See the following topics for more information:

- License Management Using the Software Product License Management Form on page 60
- Using Different Application Tiers on page 62

Note: See the *Cadence License Manager* for information regarding license configuration and management.

License Management Using the Software Product License Management Form

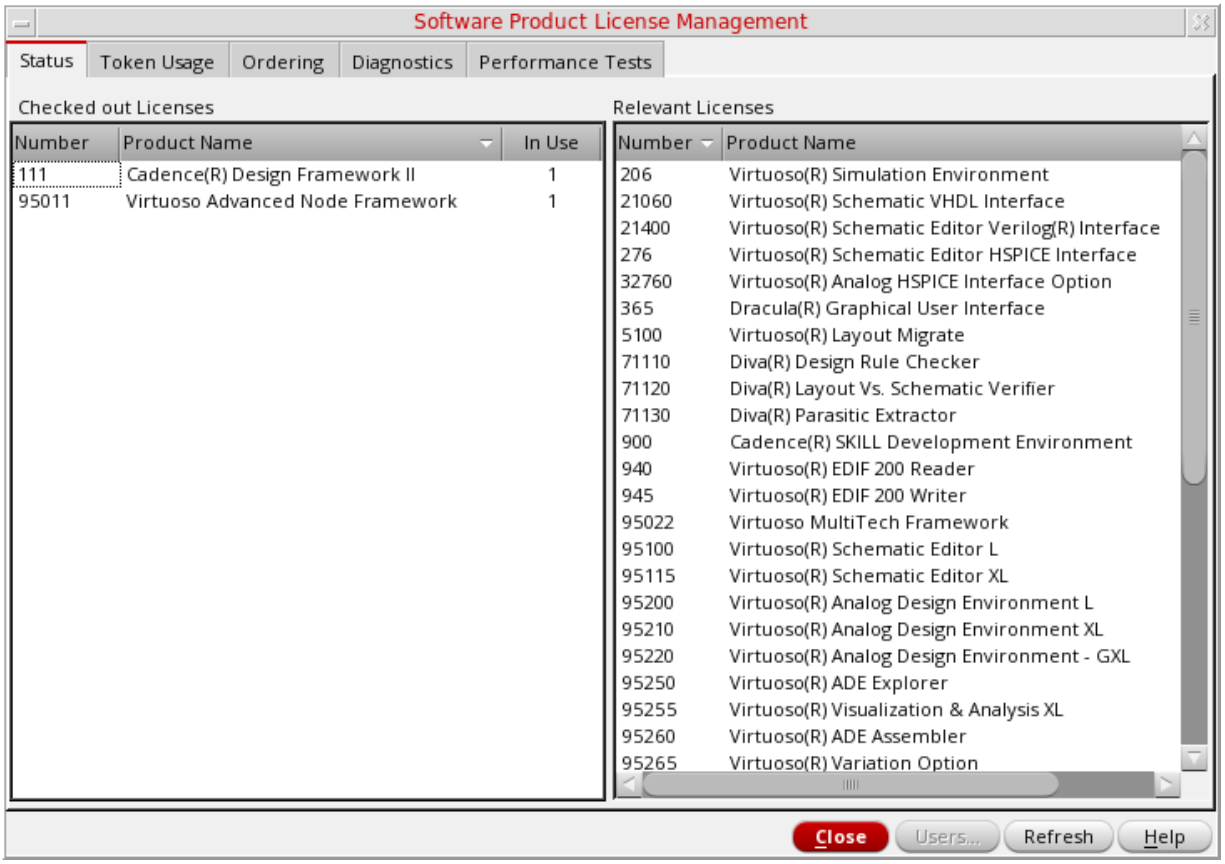
Important

See also Tracking Token Licenses in the *Virtuoso Software Licensing and Configuration Guide*.

Tracking token license activity is available in the Software Product License Management form.

To open the Software Product License Management form

1. Choose *Options* in the CIW.
2. Choose *License* from the menu.



Software Product License Management Form: Panes

- *Checked Out Licenses* – These are licenses currently running on your desktop. This pane updates when licenses are checked out by the applications or are started in the *Task Assistant* menu in the editing environments.
- *Relevant Licenses* – These are licenses available with the platform that is running.

Software Product License Management Form: Option Buttons

- *Users* shows token license usage and availability of the license you highlight. Update this form by choosing the license in either pane and clicking the *Users* button. `Control space` deselects highlighted licenses.
- *Token Information* provides details of the various capabilities of the selected application; including information on the features in use, the current availability of tokens, and the number of tokens required to run each capability.

For more information, see *Software Product License Management* form.

Using Different Application Tiers

Some Virtuoso applications are available at two or three different levels.
The most basic read-only tier is Viewer, unassisted tier is L; the standard, assisted tier is XL; the advanced, automated tiers are GXL and EXL. Each higher tier provides additional features with more automated design assistance and you can also set the default application level for a cellview.

The following table shows the tiers for various Virtuoso applications:

Application Family	Tiers
Schematic	L, XL
Layout	XL, EXL
Maestro	Explorer, Assembler
MADE	ADE, Maestro

Note: The file *your_install_dir*`/share/license/products.dfII` contains product numbers and feature strings for Virtuoso software licenses.

You can check out and use applications of different levels together. The software always checks out the highest level license you need. If you descend in your design hierarchy, the software maintains the application level you are using. When you check out an EXL application in Layout or a GXL application in ADE, the software allocates some number of flexible tokens so that you can use the advanced features of that application.

If your attempt to check out a particular license level fails, the program asks whether you want to check out the next “higher-tiered” license. For example, if you cannot check out a license for Layout XL, the program asks whether you want to check out a higher-tiered license to run the Layout application. Your answer can be *Yes*, *No*, *Always*, or *Never*. If you answer *Yes* or *Always*, the program tries to check out Layout XL first, then Layout EXL, if it fails to check out a Layout XL license. If you answer *Always*, the program automatically tries for a higher-tiered license without asking you again. If you answer *Never*, the program never tries for a higher-tiered license and never asks you again. The program writes your choice to your `.cdsenv` file. You can specify or change your preferences in `.cdsenv`.

For L and XL applications, such as Schematic and ADE, you can check out licenses one per user, host, display. You can open more than one session with a single license.

For EXL applications in Layout and GXL applications, such as Schematic and ADE, the software allocates tokens for the features you use. For more information on tokens, and licensing in general, see the *Virtuoso Software Licensing and Configuration Guide*.

Resetting Undo Stacks

To reset the undo stacks on descending into the design hierarchy, set the following environment variable:

```
designEditor.hierarchy resetUndoOnDescending 'boolean {t|nil}
```

For example:

```
(envSetVal "designEditor.hierarchy" "resetUndoOnDescending" 'boolean
t)
```

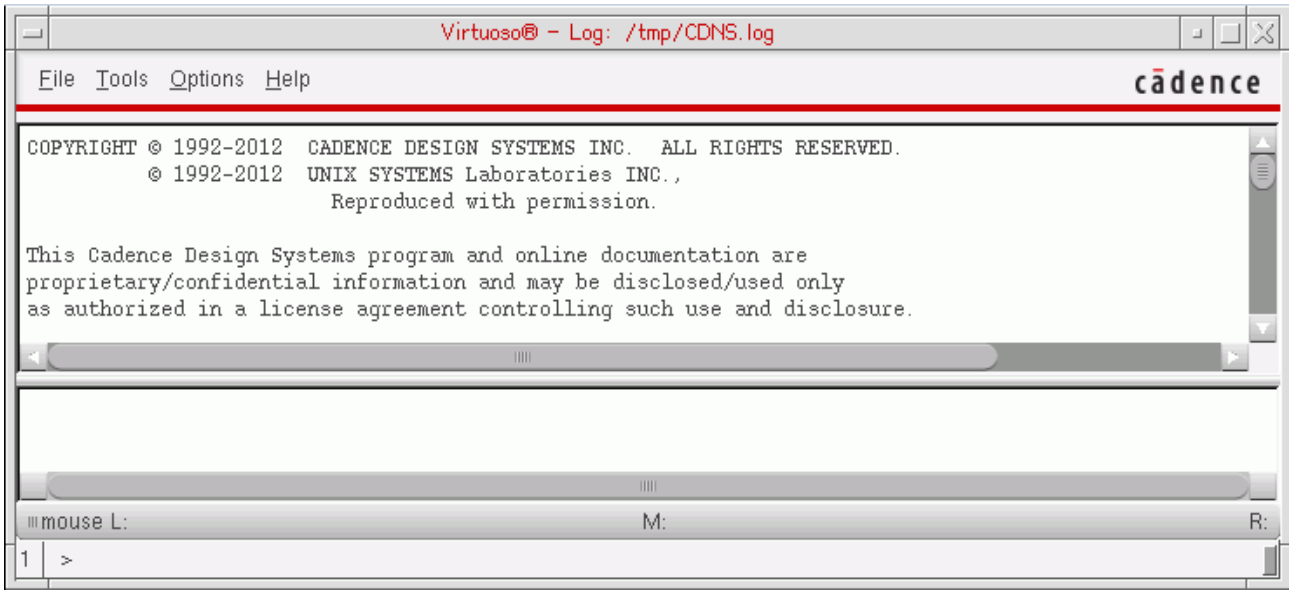
Default value: `t`

By default, the value is set to `t`. If you set the environment variable to `nil`, the undo stack will not reset while descending in the design hierarchy.

4

Using the Command Interpreter Window

Once you start Virtuoso software, you are in the Virtuoso Design Environment. You interact with Design Environment from the Command Interpreter Window (CIW). The CIW controls your Design Environment session.



The title bar of the CIW contains the following information:

- The name of the workbench you are running (for example, `virtuoso`)
- The path to the log file (`CDS.log`) that records the ongoing events of the design session

  The content of the log file is displayed in the output area.

See the following sections for information about what you can do in the CIW:

- Using the File Menu
- Using the Tools Menu
- Using the Options Menu
- Using Tear-Off Menus
- Getting Help
- Viewing Session Commands and System Responses
- Working with SKILL Commands
- Using the Mouse Bindings Line
- Viewing Required Actions on the Prompt Line
- Using the Mouse in the Virtuoso Design Environment

See also Appendix A, “Bindkeys and Access Keys.”

Using the File Menu

The following commands are available on the *File* menu:

New	Displays a submenu of the following items: ■ *Library* displays the New Library form See also "Creating a New Library in the CIW" in the *Cadence Library Manager User Guide*. **Note:** Alternative New Library forms will be displayed dependent upon whether the form was accessed from the CIW or directly from the *Cadence Library Manager*. For more information see Creating a Library in Library Manager in the *Cadence Library Manager User Guide*). ■ *Cellview* displays the New File Form See also Creating a New Cellview in the *Cadence Library Manager User Guide*
Open	See Opening a Cellview
Import	Displays a submenu of available translators on your system (see Importing Designs)
Export	Displays a submenu of available translators on your system (see Exporting Designs)
Refresh	Refreshes the design data and the CDF data. See Refreshing the View in the *Cadence Library Manager User Guide*
Make Read Only	See Making Cellviews Read-Only in the *Cadence Library Manager User Guide*
Bookmarks	Displays a list of recently viewed files (design views). For more information see Chapter 11, "Using Bookmarks and Views"

Recently Viewed Design List	Displays a list of the most recently viewed designs for quick re-opening.
	
	The format is to display in a library/cell/view format with two consecutive spaces between each to aid clarity.
	
	You can control the number of recently viewed files in this list using the `fileHistoryLimit` environment variable in the `.cdsenv` file.
	
	`ddserv fileHistoryLimit int <number_of_files>`
	
	You can set the `enableFileHistoryCheck` to `t` to check layout/cell/view before an item is placed in the history section of a File menu. The default is `nil`, which means that
	
	`ddserv enableFileHistoryCheck boolean { t | nil }`

Close Data	See "Closing All Cellviews" on page 118
Exit	See "Quitting the Cadence Software" on page 52

Using the Tools Menu

The *Tools* menu provides access to commands to start the Library Manager and your design software and utilities. The commands available on this menu will vary with the binary you are running. Some commands may include the following:

Library Manager	Opens the Library Manager (see the *Cadence Library Manager User Guide*)
Library Path Editor	Opens the Library Path Editor (see the *Cadence Library Path Editor Guide*)
NC-Verilog	Opens the Virtuoso Verilog Environment for NC-Verilog (see the *Virtuoso NC Verilog Environment User Guide*)
VHDL Toolbox	Opens the VHDL Toolbox (see the *Virtuoso VHDL Toolbox User Guide*)
Mixed Signal Environment	*Prepare Library for MSPS* displays the Create msps views & auLvs CDF simInfo form.
ADE Assembler	Launch ADE Assembler by creating a new view or opening an existing view.

ADE Explorer	Launch ADE Explorer by creating a new view or opening an existing view.
ADE Verifier	Launch ADE Verifier by creating a new view or opening an existing view.
VIVA XL	Launch Virtuoso Visualization and Analysis XL window to plot and analyze the simulation results (see the *Virtuoso Visualization and Analysis XL User Guide*)
AMS	Displays a submenu of commands for setting options and netlisting for the Cadence AMS simulator (see the *Virtuoso AMS Environment User Guide*)
Behavioral Modeling	Validates differences in measured and simulated behavior and interfaces of reference (for example, design) and compared (for example, model) blocks (see the *AMS Design and Model Validation User Guide*)
Technology File Manager	Opens the Technology File Tool Box window that lets you compile, dump, and edit technology data (see the *Virtuoso Technology Data User Guide*)
Display Resource Manager	Opens the Display Resources Tool Box (see “The Display Resources Manager and User Interface“ in the (*Technology File and Display Resource File User Guide*)
Abstract Generator	Opens the Abstract Generator form (see “Launching the Integrated Abstract Generator” in the Virtuoso Abstract Generator User Guide)
Print Hierarchy Tree	Displays a list showing the hierarchy of cellview instances in the selected cellview (see the *Virtuoso Layout Suite XL: Basic Editing User Guide*)
Set Cell Type	Displays the Set Cell Type form (see *Virtuoso Layout Suite XL: Connectivity Driven Editing*).
CDF	Displays a submenu of commands that operate on Component Description Format (see the *Component Description Format User Guide*)
SKILL IDE	Opens the Cadence SKILL IDE window (see *Cadence SKILL IDE User Guide*)

SKILL API Finder	Opens the SKILL Finder window (see *Cadence SKILL IDE User Guide*)
Conversion Tool Box	Opens the Conversion Tool Box.
Uniquify	Displays the Uniquify form

Using the Options Menu

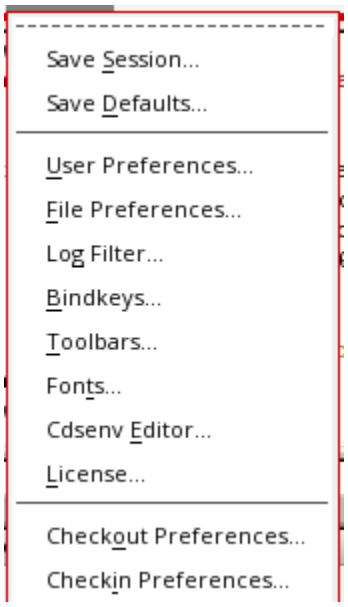
The *Options* menu provides access to the following commands for customizing your environment:

Save Session	Opens the Save Session form (see Saving Window Positions)
Save Defaults	Opens the Save Defaults form (see Specifying New Default Values)
User Preferences	Opens the User Preferences form (see Specifying User Preferences)
File Preferences	Opens the File Preferences form (see Specifying , Library Browser, and CIW Preferences)
Log Filter	Opens the Set Log File Display Filter form (see "Changing the Log Filter Options" on page 148)
Bindkeys	Opens the Bindkey Configuration form (see Configuring Application Bindkeys)
Fonts	Opens the Set Fonts form (see Viewing the Font List)
Cdsenv Editor	Opens the Cdsenv Editor form (see Using the Cdsenv Editor)
License	Opens the Software Product License Management form (see License Management Using the Software Product License Management Form)

Checkout Preferences	Opens the Auto Checkout Preferences form (see "Controlling Automatic Check-Out Behavior" in the *Cadence Library Manager User Guide*)
Checkin Preferences	Opens the Auto Checkin Preferences form (see "Controlling Automatic Check-In Behavior" in the *Cadence Library Manager User Guide*)

Using Tear-Off Menus

You can tear off menus in the Virtuoso Design Environment to have immediate access to common menus and submenus on your desktop. When you tear off a menu, the menu name appears in the title banner of the resulting menu window.



See the following topics for more information:

- Tearing Off a Menu on page 74
- Moving a Menu Window on page 75
- Resizing a Menu Window on page 75
- Maximizing a Menu Window (Linux/KDE) on page 75
- Shrinking a Menu Window to its Title Bar Only (Linux/KDE) on page 76
- Making a Menu Window Appear Always in the Foreground (Linux/KDE) on page 77
- Sending a Menu Window to Another or All Desktops (Linux/KDE) on page 77
- Configuring Menu Window Behavior (Linux/KDE) on page 77
- Closing a Menu Window on page 78
- Using the Control Menu in a Menu Window on page 78

See also

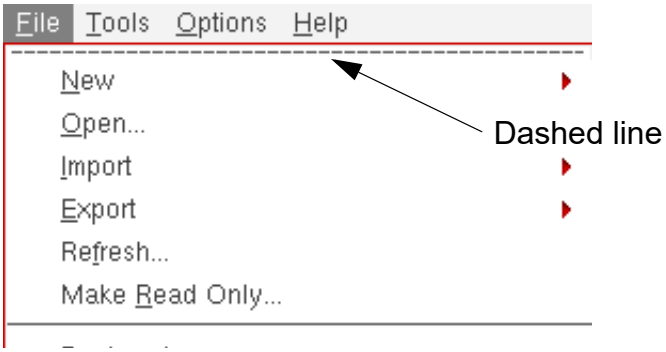
- *Tear-Off Menus* on the User Preferences form
- `tearOffMenus` setting in your `.cdsenv` file
  ("Specifying Window Controls in .cdsenv" on page 197)

Tearing Off a Menu

To tear off a menu, do the following:

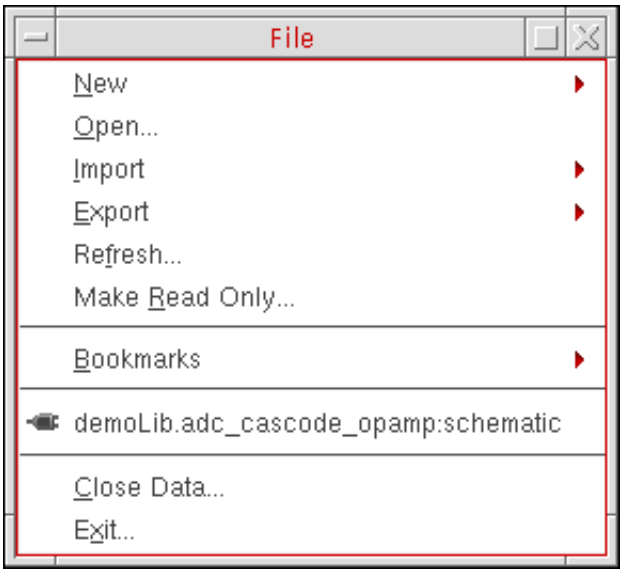
1. Select a menu, for example *File* in the CIW.

   The menu appears. There is a dashed line along the top of the menu.



2. Click the dashed line along the top of the menu.

   The menu appears in a window on your desktop. The menu name appears in the title banner of the menu window.



   You can access items on this menu directly from your desktop. You can also continue to access this menu from the menu banner in your application window.

   You can control aspects of a menu window using the control menu and other controls in the title bar.

Moving a Menu Window

To move a menu window, do the following:

➤ Click the mouse anywhere in the title bar of the menu window and drag-and-drop the menu window to the location you want.

Alternatively, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *Move*.

   The menu frame (outline) appears beneath your cursor.

3. Move the menu window to the location you want and click to place it.

Resizing a Menu Window

To resize a menu window, do the following:

➤ Hover the cursor over any corner of the menu window until the sizing cursor appears, then drag-and-drop to set the size (width and height).

Note: Whether you can resize a menu or not depends on the settings and version of the window manager that you are using.

Alternatively, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select the size/resize option (for example, *Size* on Linux running KDE or *Resize* on Solaris).

   The menu frame (outline) appears along with a sizing cursor ( ).

3. Drag-and-drop any corner of the menu window to set the size (width and height).

Maximizing a Menu Window (Linux/KDE)

To maximize a menu window, do the following:

➤ Click the square in the upper right corner of the menu window.

The menu window fills your screen.

➤ Click the square again to restore the menu window to it previous view.

Note: Whether you can see the square button or not depends on the settings and version of the window manager that you are using.

Alternatively, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *Maximize*.

   The menu window fills your screen. A check mark appears to the left of of the *Maximize* entry on the control menu.

   You can restore the menu window to its previous view by selecting *Maximize* from the control menu again.

Shrinking a Menu Window to its Title Bar Only (Linux/KDE)

To shrink a menu window to its title bar only, do the following:

➤ Double-click the title bar.

Only the title bar of the menu window is visible.

➤ Double-click the title bar again to restore the menu window to it previous view.

Alternatively, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *Shade*.

   The menu window shrinks to its title bar only. A check mark appears to the left of the *Shade* entry on the control menu.

   You can restore the menu window to its previous view by selecting *Shade* from the control menu again.

Making a Menu Window Appear Always in the Foreground (Linux/KDE)

To make a menu window appear always in the foreground of your display, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *Always on Top*.

   The menu window stays in the foreground.

Sending a Menu Window to Another or All Desktops (Linux/KDE)

If you have more than one desktop, you can send a menu window to another desktop or to all desktops.

To send a menu window to another or all desktops, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *To Desktop*.

   A side menu appears.

3. From the side menu, select a desktop or select *All Desktops* to send the menu window to all desktops.

   If you selected another desktop, the menu window appears on that desktop (only).

   If you selected *All Desktops*, the menu window appears on all your desktops.

Configuring Menu Window Behavior (Linux/KDE)

To configure menu window behavior, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *Configure Window Behavior*.

   The Settings – KDE Control Module window appears

3. Use this configuration window to change the look and feel of the menu window.

4. Click *OK*.

Closing a Menu Window

(Linux only) To close a menu window, do the following:

➤ In the upper right corner of the window, click the *X*.

The menu window closes.

(Linux and Solaris) To close a menu window, do the following:

1. In the upper left corner of the window, click the dash (-).

   The control menu appears.

2. From the control menu, select *Close*.

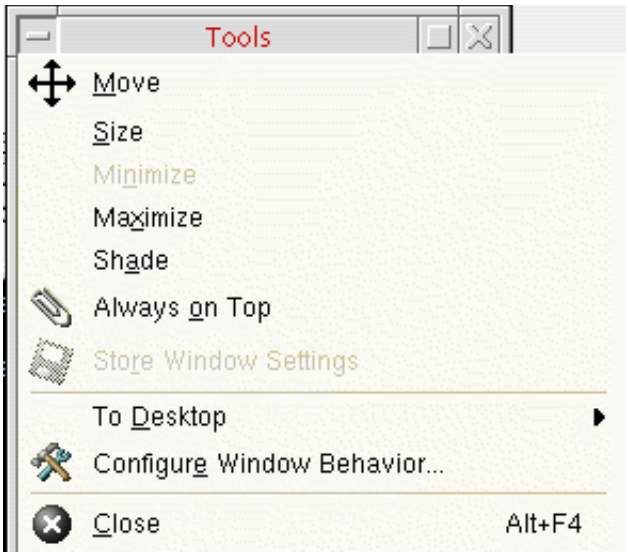
   The menu window closes.

Using the Control Menu in a Menu Window

To control aspects of a menu window, do the following:

1. In the upper left corner of the window, click the dash (-).

The control menu appears. The items that appear on this menu depend on your operating system and window manager. The following menu appears under Linux running KDE:



2. From the control menu, select the action you want to perform.

   For example, under Linux running KDE, you can select one of the following:

Move

The menu frame (outline) appears so that you can move it to the location you want and click to place it.

Tip

Alternatively, you can drag-and-drop a menu window to the location you want.

Size

The menu frame (outline) appears along with a sizing cursor ( ) so that you can drag-and-drop any corner to set the size (width and height).

Tip

Alternatively, you can hover the cursor over any corner of a menu window until the sizing cursor appears, then drag-and-drop to set the size.

Maximize

The menu window fills your screen. A check mark appears to the left of of the *Maximize* entry on the control menu. You can restore the menu window to its previous view by selecting *Maximize* again.

Tip

Alternatively, you can click the square in the upper right corner of the menu window to maximize it. Click the square again to restore the menu window to it previous view.

Shade

The menu window shrinks to its title bar only. A check mark appears to the left of the *Shade* entry on the control menu. You can restore the menu window to its previous view by selecting *Shade* again.

Tip

Alternatively, you can double-click the title bar to toggle this view.

Always on Top

The menu window stays in the foreground.

To Desktop

If you have more than one desktop, this action sends the menu window to the selected desktop or to *All Desktops*.

Configure Window Behavior

The Settings – KDE Control Module window appears so that you can change the look and feel of the menu window.

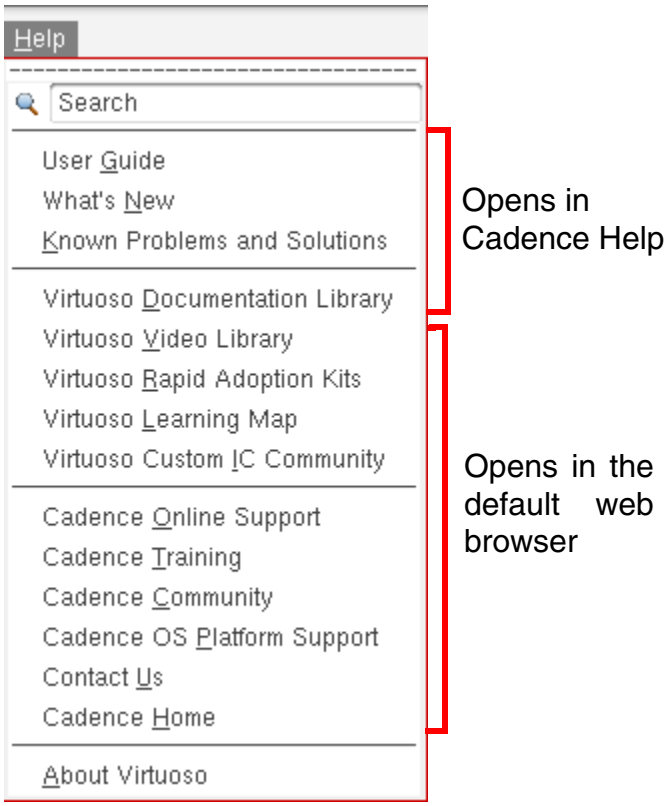
Close

The menu window closes.

Getting Help

For help with various Virtuoso products, do the following:

1. Choose *Help*.



2. Choose one of the following menu options:

Help menu option	Description
Search	A text field that lets you enter a search string. Press `Enter` to view the search results. **Note:** Do not enclose the search string in double quotes.
User Guide	Opens Virtuoso Design Environment User Guide (at the section that provides information about using CIW) in Cadence Help.
What's New	Opens the Virtuoso What's New document in Cadence Help.
Known Problems and Solutions	Opens the Virtuoso Known Problems and Solutions document in Cadence Help.

Virtuoso Documentation Library	Opens the Cadence Help home page, which provides quick access links to the following local and online resources: ■ What's New ■ Video Demos and Tutorials ■ Featured Content ■ Known Problems and Solutions ■ Other web resources
Virtuoso Video Library	Opens the Video Library page available on Cadence Online Support (COS). This page lists the videos available for various Virtuoso products. **Note:** You must have a COS account to access the content available on COS. **Note:** Contact your IT support to ensure that the Internet ports required for video playback are enabled.
Virtuoso Rapid Adoption Kits	Opens the Rapid Adoption Kits page on COS. This page lists Rapid Adoption Kits (RAKs) available for various Virtuoso products. A RAK contains design databases and instructions on how to run the design flow.
Virtuoso Learning Map	Lists domain-specific training available on Cadence Training Services. Cadence Training Services learning maps provide a comprehensive visual overview of the learning opportunities for Cadence customers. They provide recommended course flows as well as tool experience and knowledge levels to guide customers through a complete learning plan.
Virtuoso Custom IC Community	Opens the Virtuoso Custom IC Community web page. This page provides access to the latest blogs and discussion threads on various Virtuoso products and design topics, information about software downloads and support and training, and other related information. You too can contribute to the community forum by creating a Cadence account. This gives you additional benefits such as alerts about topics of interest and access to online webinars.

Cadence Online Support	Opens COS, which you can use to access information about Cadence products, documentation, videos, RAKs, application notes, troubleshooting information, alerts, and so on. Improvements are regularly made to COS to make it easy for you to look up the information you want. We recommend that you bookmark this web site and use it as your first point of reference for any Virtuoso-related information.  *Tip* You can also access COS by clicking the Cadence logo available in the upper-right banner in each Virtuoso window.  To disable this behavior, you need set the following shell variable: `Setenv CDS_LOGO_ACTION OFF`
Cadence Training	Opens the Cadence training web page. You can find on this page information about the training courses available in different regions. Information is available about both classroom and online courses.
Cadence Community	Opens the Cadence Community web page. This page provides access to the latest blogs and discussion threads on various Cadence products and solutions, and EDA Industry Insights. You too can contribute to the community forum by creating a Cadence account. This gives you additional benefits such as alerts about topics of interest and access to online webinars.
Cadence OS Platform Support	Provides information about the current Cadence software releases and the supported platforms.
Contact Us	Opens the Cadence Customer Support web page, which provides customer support contact information for different regions.
Cadence Home	Opens the Cadence corporate web site.
About Virtuoso	Displays Virtuoso Design Environment version information.

Viewing Session Commands and System Responses

You can view the commands you use and the responses the system makes during a design session in the output area of the Command Interpreter Window (CIW). These commands and responses are in SKILL code. As well as displaying this information in the output area, the software also writes it to the `CDS.log` file.

The output area normally displays only a few lines of code. You can resize the window or use the scroll bars to view more lines. When you resize the CIW, only the output area changes size. You can jump to the top or to the end of the output area by typing *Ctrl*+*Home* or *Ctrl*+*End*. You can see the full record of activity by opening the `CDS.log` file in a text editor.

Note: You can control the maximum number of previous commands that appear in the output area (see "Specifying CIW Controls" on page 136).

You can control what you see in the output area by changing the log filter (see "Changing the Log Filter Options" on page 148). By default, output is limited to error messages, warning messages, program results, and results of running a function. The log file contains a complete record of the session including menu commands, prompts, and other kinds of output. If you want to see this additional information, you can set the log filter options to change the display.

Note: The prefix `\#` is used in the `CDS.log` file to filter out statistic information such as output memory use and X resource ID.

Working with SKILL Commands

CIW lets you interact with Virtuoso SKILL commands. The following sections describe the different interactions you can have with SKILL commands in CIW:

- Typing SKILL Commands
- Executing SKILL Commands
- Debugging Commands
- Repeating Commands from the CIW

Typing SKILL Commands

You can type a SKILL command in the edit area of the Command Interpreter Window (CIW). Edit area is the area in the CIW below the displayed history. For example:

1. To open the Library Manager from the CIW, type the following in the edit area of the CIW:

   ```
   ddsOpenLibManager
   ```

2. Press the `Return` or `Enter` key.

   The Library Manager form appears. By default, pressing the `Return` or `Enter` key executes the command you type.

To insert a new line / command, press any of the following key combinations in the edit area: `Shift+Return`, `Shift+Enter`, `Ctrl+J` or `Ctrl+M`.

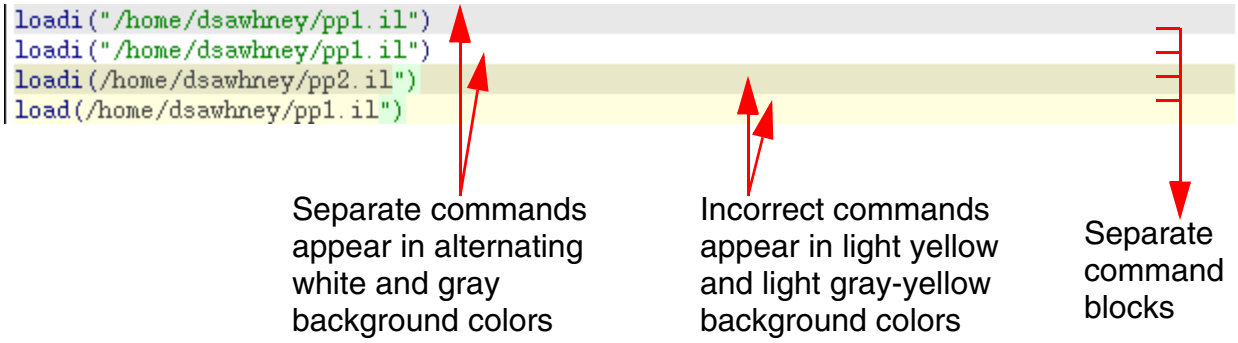
However, the bindkeys (or key combinations) you use in CIW to type commands work in conjunction with the CIW controls you select on the User Preferences form. For example, if in the *CIW Controls* section on the User Preferences form, you clear the *Enter Key Executes Command* check box, which is otherwise selected by default, then the `Enter` key would insert a new line instead of executing the command you type, and you would use `Shift+Enter` to execute the command.

Tip

You can specify the initial number of lines (size) of the input area as well as the number of previous commands that appear in the input area (see "Specifying CIW Controls" on page 136).

Executing SKILL Commands

Starting from this release, separate commands appear with alternating white and gray background colors. Each color separated command is referred to as a command block. The CIW syntax checker checks the syntax of the command entered in a command block and provides immediate feedback. Incorrect or incomplete commands are indicated with light yellow or light gray-yellow background colors, respectively.



To execute a command block, press `Enter`. When a command is executed, the complete command block is sent to the SKILL parser for further syntax checking.

Note: Unlike the SKILL parser, the CIW syntax checker only checks the “syntax complete” status of SKILL commands. Commands are considered syntax complete if they do not contain unmatched parentheses, braces, brackets, double quotes or multi-line comments. Syntax complete commands need not necessarily be syntax correct.

You can also select and execute partial commands. To select a partial command, use the key combination `Shift+Ctrl+Q`. The key combination `Shift+Ctrl+Q` selects the nearest command expression if no commands are currently selected in a command block. If the selection is within a command block, `Shift+Ctrl+Q` extends the selection to include more of the command block, first upwards, and then downwards, until the entire command block is selected.

Note: If commands from multiple command blocks are selected, the key combination `Shift+Ctrl+Q` does no further selection.

To execute a selected command, use the key combination `Alt+Enter` or `Alt+Return`. If, however, the selected text contains incomplete or incorrect syntax, the syntax will only be copied to the edit area as text.

For a detailed list of bindkey combinations that you can use to select and execute commands, see “Appendix C, Bindkeys and Access Keys”.

Debugging Commands

Starting from this release, the SKILL parser provides immediate feedback and helps debug the syntax with the following features:

- Matching parenthesis (`()`), brackets (`[]`), and braces (`{}`)

```
defmethod(printself ((obj fixnum))
  sprintf(nil "FIXNUM{%d}" obj))
```

Matching parenthesis are highlighted in blue

- Using the right bracket (`]`) to close all open parentheses within a command block. The right bracket does not balance open parenthesis beyond the selected command block.

```
defmethod(printself ((obj fixnum))
  sprintf(nil "FIXNUM{%d}" obj))
printf("Percentage A prints %A\n" 42
printf("use L %L\n" 42)]
```

Right bracket closes any or all open parenthesis

- Displaying incorrect commands with light yellow or light gray-yellow background colors.

```
defmethod(printself ((obj fixnum)
  sprintf(nil "FIXNUM{%d}" obj))
```

Incorrect command is displayed with a light-yellow background color

- Providing customizable colors for various syntax elements
  You can customize the foreground and background colors of various syntax elements by using the `envSetVal()` function or specifying them in your `.cdsenv` file at any time. The various syntax elements are as follows:

`Syntax Element`	Description

`ciwCommentColor`	Used for single line and multi-line comments Single line comments begin with a semicolon (`;`) and multi-line comments are surrounded by "`/*`" and "`*/`"
`ciwParenColor`	Used for parentheses (`()`)
`ciwBracketColor`	Used for square brackets (`[]`)
`ciwBraceColor`	Used for curly braces (`{ }`)
`ciwSymbolColor`	Used for explicit symbols and for numeric ranges, such as "`0:10`" or "`3.4:4.8`" Explicit symbols are words preceded by a single quote (`'`)
`ciwKeywordColor`	Used for the known set of SKILL keywords, such as `abs`, `acos`, `cdar`, `cddr`, and so on
`ciwStringColor`	Used for strings enclosed within double quotes
`ciwMatchParenColor`	Used for the background color to highlight the parenthesis, brace or bracket next to the text cursor or that matches the nearest one **Note:** If there is a parenthesis, brace or bracket on either side of the text cursor, the one to the left of the cursor is highlighted. If there is no match (or if the parenthesis, brace or bracket is within a string or comment) nothing is highlighted.
`ciwMismatchParenColor`	Used as the background color to highlight mismatched parentheses, braces and brackets. Mismatched colors are highlighted only on the closing side. If there are too many opening parentheses, braces or brackets, they will not be highlighted as mismatched instead the command area background will be set to one of the mismatched command colors. If there are too many closing parentheses, braces or brackets within a command block, the first unmatched closing parenthesis, brace or bracket will be highlighted with this color, and any subsequent parentheses, braces and brackets will also be highlighted with this color within the same command block until balance is restored.
`ciwUnmatchQuoteColor`	Used as the background color for strings that do not have a closing double quote, or that start and end on different lines

`ciwMismatchCmdColor1`	Used to indicate an incomplete syntax status for the command blocks that have a white background Syntax is said to be incomplete if it contains unmatched parentheses, braces, brackets, double quotes or multi-line comments. **Note:** Syntax complete commands are not necessarily syntax correct.
`ciwMismatchCmdColor2`	Same as `ciwMismatchCmdColor1` only that this is used for command blocks that have a light gray background

The default setting for all colors is as follows:

```
ui    ciwCommentColor        string "#800000"
ui    ciwParenColor        string "#191970"
ui    ciwBracketColor        string "#000080"
ui    ciwBraceColor        string "#0000A0"
ui    ciwSymbolColor        string "#400080"
ui    ciwKeywordColor        string "#000080"
ui    ciwStringColor        string "#008000"
ui    ciwMatchParenColor    string "#80dcff"
ui    ciwMismatchParenColor    string "red"
ui    ciwUnmatchQuoteColor    string "#e0ffe0"
ui    ciwMismatchCmdColor1    string "#ffffe0"
ui    ciwMismatchCmdColor2    string "#e8e8c8"
```

Repeating Commands from the CIW

You can use the following methods to quickly review, repeat, and/or rerun commands in the CIW:

- To scroll through the previously entered CIW commands use your keyboard arrow keys.

That is, if you have *directly (text) entered* commands into the CIW you can recall them, and then rerun them, using the `Up` and `Down` arrow keys, followed by pressing the `Enter` key.

- To rerun a command that starts with a specified pattern (as per the CIW screenshot above), for example:

  ```
  236 hiFormCancel(schFormOfEditorOptions)
  ```

  enter:

  ```
  !hiFo
  ```

  This will execute the last command whose prefix *starts* with `hiFo` in the CIW history.

- To rerun a command that *contains* a specified pattern, enter:

  ```
  !?pattern
  ```

  For example, if you enter "!?`iFo`" this will look for commands that contain "`iFo`", such as `hiFormCancel`.

- To rerun the *last* command entered in the CIW, enter:

  ```
  !!
  ```

- To copy and paste a command, double-click to select a word in the CIW output area or triple-click to select a complete command line. Paste that word or line into the CIW input area with either a drag-and-drop action or by using the middle-mouse-button. Optionally, edit the command as required. Press the `Return` key to action the command.

You can re-size (enlarge) the CIW Command editarea to displaymore commands that have been entered.

Note: Although the complete history of all types commands is visible at all times, however, if on the User Preferences form in the *CIW Controls* section, you select the *Retain Unique Commands* option, only unique commands are retained. All previous identical commands are deleted.

Using the Mouse Bindings Line

Mouse bindings are mouse button settings that are assigned, or bound, to SKILL commands. Mouse bindings tell you what command the program executes when you press a mouse button. You can see the current mouse bindings for the window where the pointer is on the mouse bindings line. For example:

The letters L, M, and R indicate the left, middle, and right button assignments.

You can perform the following tasks associated with mouse bindings:

- Control whether the mouse bindings line appears in your CIW using the *MouseBar* check box on the pop-up that appears when you right-click the mouse bindings line.
- Control whether the mouse bindings line appears in new windows using the *Mouse Prompts* drop-down combo box in the *Window Controls* group box on the User Preferences form.

  Note: The mouse bindings setting is overridden by any applied workspace regardless of the position selected in the Mouse Prompts combo-box.
- Mouse bindings display changes to show the appropriate commands when modifier keys on the keyboard are pressed in combination with a mouse click.

  In case a mouse plus a keyboard modifier binding, such as mouse click + <SHIFT>, is not working, it is likely that you are using an older VNC server that is not compatible with the KeyCode to KeySym conversion function `XkbKeycodeToKeysym` that replaces the deprecated function `XKeycodeToKeysym`.

  To workaround this issue on older VNC servers, use deprecated the `XKeycodeToKeysym` function. Also, update the event list to include new `GenericEvent`.

For more information, see Appendix A, “Bindkeys and Access Keys”.

Viewing Required Actions on the Prompt Line

When a command requires your input, the required action appears as a line of text on the prompt line at the bottom of the window. For example, if you are using the schematic editor and have selected the zoom in function, the following line of text appears on the prompt line:

```
Enter the first corner of the box you wish to enlarge.
```

Prompts appear in the CIW even though the results of the action take place in a design window.

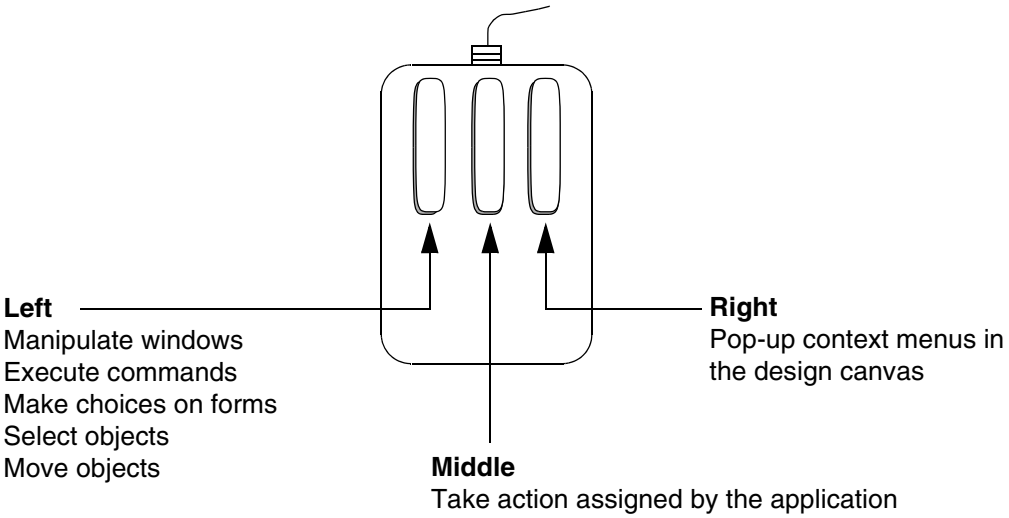
Note: A numerical window identifier (a window number) appears to the left of the prompt line. The CIW is usually window number 1 because it is the first one that appears when you start your Cadence software.

Tip

To cancel a command and remove the prompt, press the *Escape* key while the cursor is in the design window (the current, active window).

Using the Mouse in the Virtuoso Design Environment

You will use mouse buttons in the following way:



Each Virtuoso application has its own mouse button settings. You can change or add to these settings using the Bindkey Configuration form or by editing your `.cdsinit` file. You can view mouse button bindings for each application from the Bindkey Configuration form (see "Viewing the Current Bindkeys for an Application" on page 311).

Each time you issue a command, the mouse bindings line changes to show what the mouse buttons do when you use that command.

You can specify mouse button settings related to window management tasks—such as raising, lowering, resizing, and minimizing windows—in your `.Xdefaults` file. You can edit this file to add or change settings.

If your mouse has a scroll wheel, the default bindings are as follows:

[Key +] Scroll Wheel	Action	SKILL Command
Scroll wheel up	Scroll/pan up	`hiSetBindKey(p "None<Btn4Down>" "geScroll(nil \"n\" nil)")`
Scroll wheel down	Scroll/pan down	`hiSetBindKey(p "None<Btn5Down>" "geScroll(nil \"s\" nil)")`

[Key +] Scroll Wheel	Action	SKILL Command
Ctrl + Scroll wheel up	Zoom in	`hiSetBindKey(p "Ctrl<Btn4Down>" "hiZoomInAtMouse()")`
Ctrl + Scroll wheel down	Zoom out	`hiSetBindKey(p "Ctrl<Btn5Down>" "hiZoomOutAtMouse()")`
Shift + Scroll wheel up	Pan left	`hiSetBindKey(p "Shift<Btn4Down>" "geScroll(nil \"w\" nil)")`
Shift + Scroll wheel down	Pan right	`hiSetBindKey(p "Shift<Btn5Down>" "geScroll(nil \"e\" nil)")`

See also Appendix A, “Bindkeys and Access Keys.”

License Activity Indicator

The resource license activity indicator, if activated, is displayed in the status bars of the CIW and Virtuoso session windows (for example, in ADE, the Schematic Editor, and the Layout Suite), where it provides visual feedback about the current license level usage.

This information can then serve as an alert system for when Virtuoso resource usage is being stretched, for example, if high license activity is causing temporary non-responsiveness.

See also Specifying Dashboard Controls for information on controlling the display of dashboard indicators in the CIW and/or current session windows.

Note: For information on customizing display settings for the resource indicator, see Resource Indicator Environment Variables.

The resource indicator is located to the right of the status bar and displays, using a measurement light, license activity and license usage levels by tracking current access to the license server.

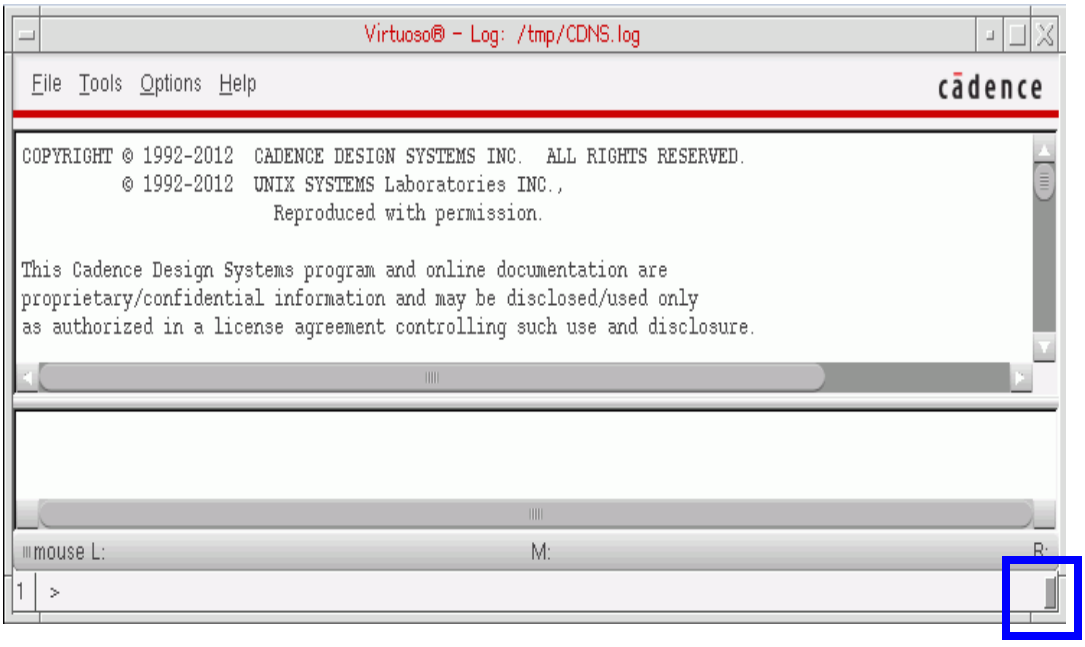


Figure 4-1 License Activity Indicator Highlighted in the CIW

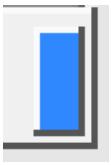


Figure 4-2 License Activity Indicator On

Resource Indicator Environment Variables

Resource (license activity) indicators can be displayed in the CIW and Virtuoso session windows. You can use the following environment variables to control their display:

- `ui dashboardIndicators boolean true/false`

Specifies whether indicators should be displayed or not. Setting to t(rue) will enable display of the resource indicators as specified by the `dashboardLicenseLight` control below.

- `ui dashboardLicenseLight cyclic <value>`

Specifies the display of the resource indicator. The default is to display in both "`CIW+Session`". Other acceptable values are "`CIW`" to only display in the CIW, or "`Hidden`" to not display the license activity indicator anywhere in Virtuoso.

Virtuoso Performance Warning Messages

Virtuoso performance warning messages are displayed depending on the following measurement criteria, and their related environment variable settings:

- Low Memory Warnings
- Swap Activity Warnings

Low Memory Warnings

Virtuoso performance warning messages, related to low memory measurements, are displayed depending on the current memory status of the system and the Virtuoso sub-version (64-bit) in relation to the following environment variable settings:

- `ui lowMemory64bWarnLevel1 float 20.0`
- `ui lowMemory64bWarnLevel2 float 10.0`
- `ui lowMemory64bWarnLevel3 float 4.0`

Legal values can be set between 0.0 and 99.0, inclusive.

You can set these levels to 0 to disable the warnings.

The environment variable with the highest value sets the threshold for the first low memory warning message. Similarly, the variables with the middle and lowest values set the threshold for the second and third warning message, respectively. Therefore, `lowMemoryWarnLevel1` does not have to be set as a higher value than `lowMemoryWarnLevel2` or `lowMemoryWarnLevel3`.

The default low memory environment variable values of `20`, `10`, and `4`, represent percentage settings of maximum memory left available in the system that is currently running Virtuoso. Therefore, once the memory usage threshold of 80% (100-20) is exceeded, the first low memory warning message is displayed, followed by a further warning at 90% (100-10) and 96% (100-4), respectively.

If a memory warning level variable value is changed from the current memory value to a new value that is equal to or greater than the current memory available, then that immediately triggers the display of a low memory warning message.

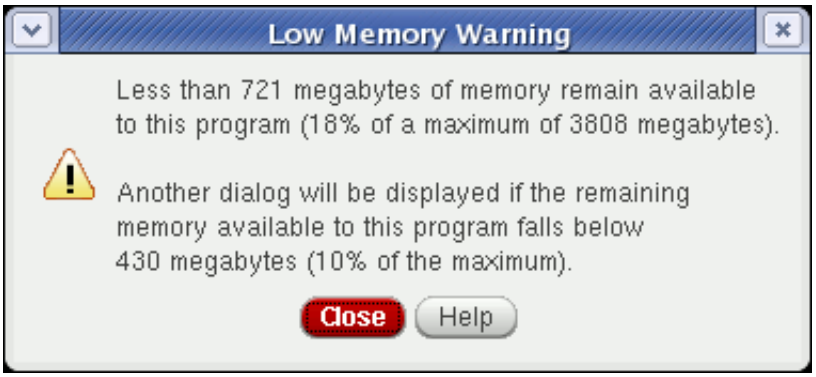
These percentage measurements are based on the theoretical maximum size that a Virtuoso process can reach on the system that it is being run on. 32-bit processes have a physical memory limit of 4 GB (although on some systems that physical maximum can be smaller than 4 GB), while 64-bit processes can theoretically be as large as the total system memory that

is available. Therefore, system memory size is usually used to calculate the theoretical maximum size for 64-bit programs.

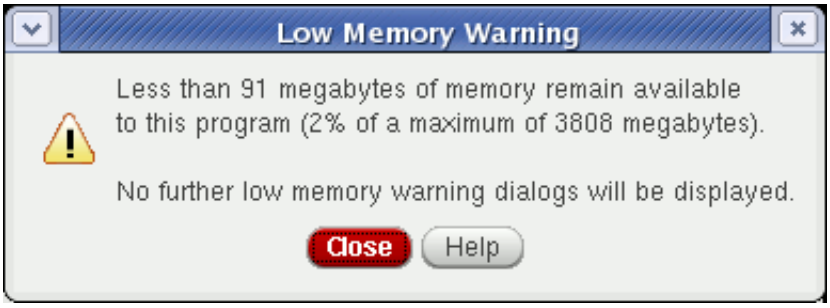
Note: You can view memory usage details, such as the theoretical maximum size for a Virtuoso process and the process size in the `CDS.log` file under *Maximum memory size* and *process size*, respectively.

Note: If the *Maximum memory size* changes by 2% or more, it is logged.

When available system memory reduces so that it reaches or crosses the highest or the middle memory warning thresholds, or both, the following warning dialog is displayed:



When the lowest memory threshold is reached or exceeded, then the following warning message is displayed:



Note the following in relation to low memory warning message display:

- Display of low memory warning messages do not log a command to the `CDS.log` file, and are not replayable.
- Comment lines are logged when the low memory warning is displayed.
- Low memory messages are not displayed in the replay mode, but the low memory comment lines are logged if any thresholds are crossed.

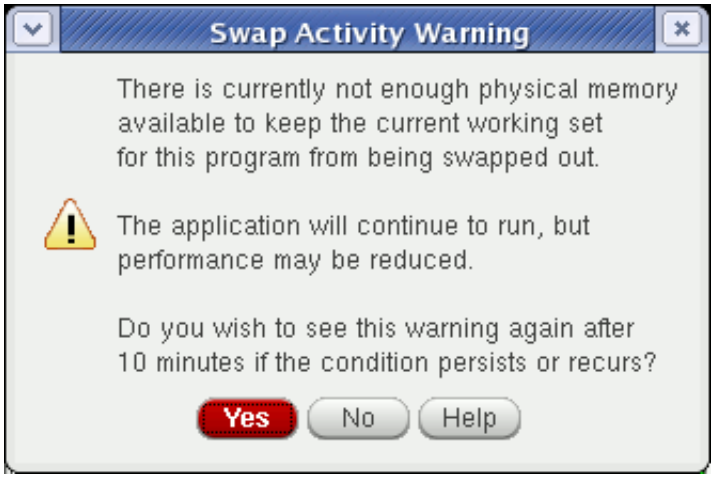
- For the graphical replay that does not cause `Virtuoso` to exit, the low memory warning messages are displayed, as appropriate, once the session enters or re-enters the interactive mode.
- In the interactive `nograph` mode, a warning message is output when low memory settings are reached or crossed.

Swap Activity Warnings

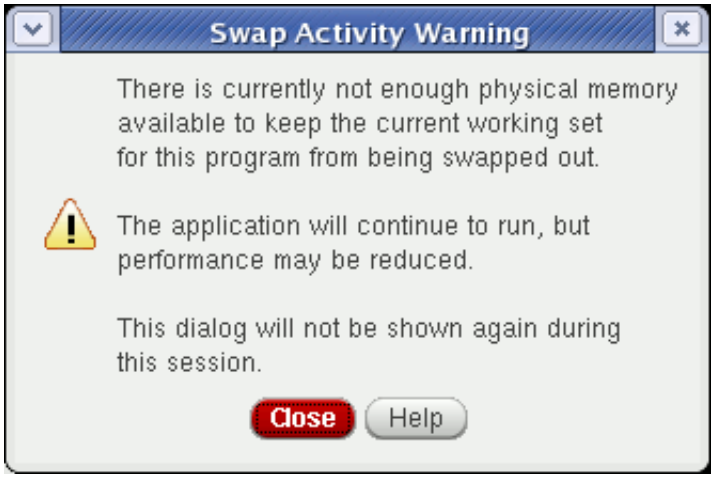
Virtuoso performance warning messages, related to page swap activity measurements are displayed dependent on the current settings of the following environment variables:

- `ui swapActivityWarnMinutes int 10`

  Controls how many minutes must elapse before the swap activity warning message are displayed again, assuming that you select *Yes* on the message window to have the message displayed again, should the problem persist. The default is `10`.



Note: If `swapActivityWarnMinutes` is set to `0`, then the *Do you want to see this dialog again* text in the warning message is replaced with *This dialog will not be shown again during this session*, and the message is displayed only one time.



- `ui swapActivityMajorPageFaults int 200`

  Controls the minimum number of *major* page faults required to occur, over one or more 10 second periods, before the swap activity warning message is triggered to display. Default is `200`.

- `ui swapActivityMinorPageFaults int 10000`

  Controls the minimum number of *minor* page faults required to occur, over one or more 10 second periods, before the swap activity warning message is triggered to display. Default is `10000`.

- `ui swapActivityWarnPeriods int 3`

  Controls the number of consecutive 10 second periods that must have passed, for the minor or major page fault count to equal or exceed its setting, before the swap activity warning message is displayed.

  Note: If `swapActivityWarnPeriods` is set to `0`, then the swap warning message will not be displayed, as the number of page faults that occur will not be checked.

- `ui swapActivityEnableMinorPageFaults boolean nil`

  Controls whether or not minor page faults will generate a swap warning message. The default is `nil`. If set to `t`, the Swap Activity Warning dialog will update to state "*The amount of memory being accessed by the current procedure is generating a large number of cache hit (minor page faults)*".

  Note: If you select *No* in this (minor page faults) dialog, to not show that warning again, it will not prevent the dialog displaying for major page faults (not enough physical memory). However, if you click *No* in the major page faults dialog, that will prevent the minor page faults dialog from displaying.

You can use the `CDS_MEMPERF_WARNLOG` shell environment variable to run user-specified executables. For example, the following command will make the executable file `/usr/local/warnIssued` to run asynchronously (in the background) and provided with the argument `SwapActivityDialog`.

```
setenv CDS_MEMPERF_WARNLOG="WHEN=SwapActivityDialog SCRIPT=/usr/local/warnIssued"
```

Output, including failures to launch the application, is displayed in the terminal window from where Virtuoso was launched.

Here,

- Value will be several pairs of name and value. Any unrecognized names are ignored.
- `WHEN`
  Specifies when the program should run.

- `WHEN=SwapActivityDialog`
  Indicates the provided script will be run when the *Swap Activity Warning* dialog box is displayed.
- Any unrecognized value of `WHEN` is silently ignored.
- `SCRIPT`
  Specifies the program which will be used and can be specified only once. The associated value will consist of one or more alpha-numeric, underscore, hyphen, period, or forward-slash characters (no whitespace or characters which are special to the shell).

Some facts about the swap activity warning messages:

- These messages do not log a command to the `CDS.log` file and are not replayable.
- Comment lines are not logged.
- The messages are not displayed in replay mode
- The swap activity comment lines are not logged if any thresholds are reached or crossed.
- If you are performing graphical replay, and the replay does not cause `Virtuoso` to exit, then the swap activity warning messages are displayed, after the session enters or re-enters interactive mode.
- In interactive `nograph` mode, a warning message is displayed when the swap activity settings are reached or crossed.

  Additionally, because it is not possible to respond to a message in interactive `nograph` mode, the message for the repeat display will state that: *Another warning will be output after X minutes if the condition persists or recurs*.

  Note: The only way to turn disable these messages is to set the appropriate environment variable accordingly.

5

Working with Cellviews

You work with cellviews in a session window. When you first open a cellview for design editing, a session window appears. You can open more than one cellview in a session window. Each cellview appears on its own tab (see "Tabs in a Session Window" on page 225).

For information about working with cellviews, see the following sections:

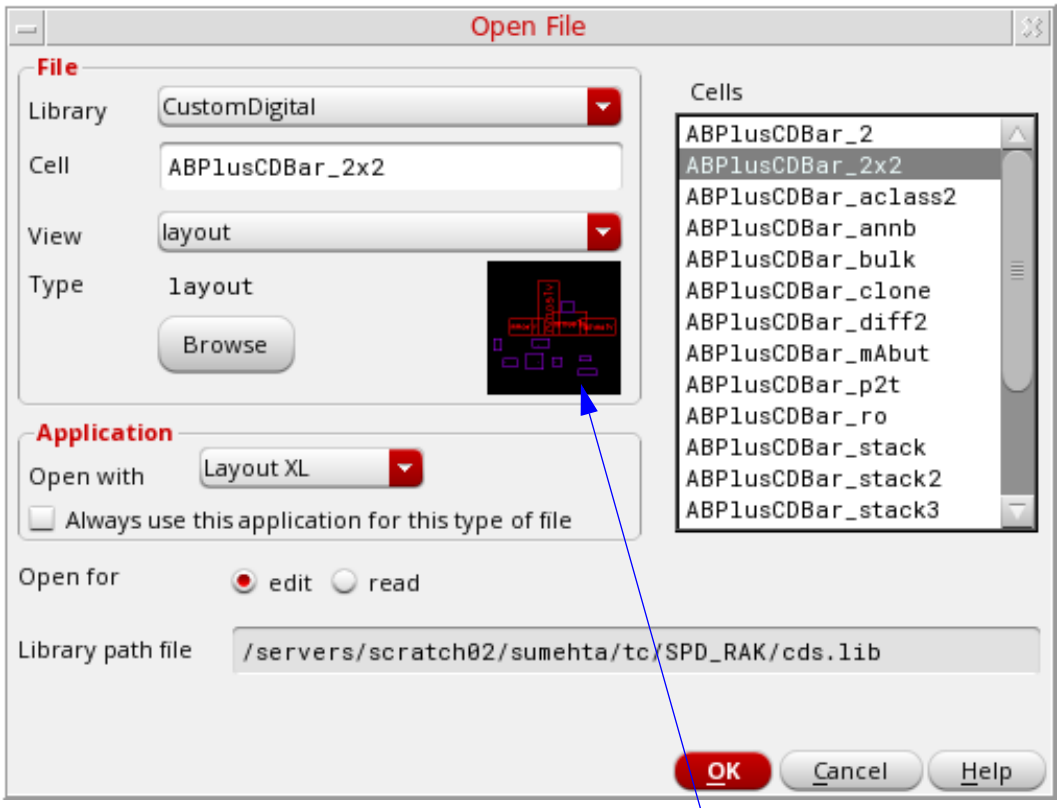
Opening a Cellview

To open a cellview from the Command Interpreter Window (CIW) or your session window, do the following:

1. Choose *File – Open*.

   Note: Alternatively, if you have a tab on display in your session window, right-click over the tab and select *Open* from the displayed pop-up menu to display the form.

   The Open File form appears.



Thumbnail of the selected cellview is displayed (if found) to provide a visual selection aid.

   Note: When you choose *File – Open* in your session window, or select *Open* from the tab pop-up menu, additional radio buttons appear near the bottom of the form so that you can choose whether to open the cellview on a new tab, on the current tab, or in a new session window (see step 5).

1. In the *File* group box, do the following:

a. In the *Library* drop-down combo box, select a library.

Cell names from the selected library appear in the *Cells* list box.

b. Select a cell name from the *Cells* list or directly type the cell name in the *Cell* field.

Note: When you type the cell name, the most relevant cell in the *Cells* list gets selected.

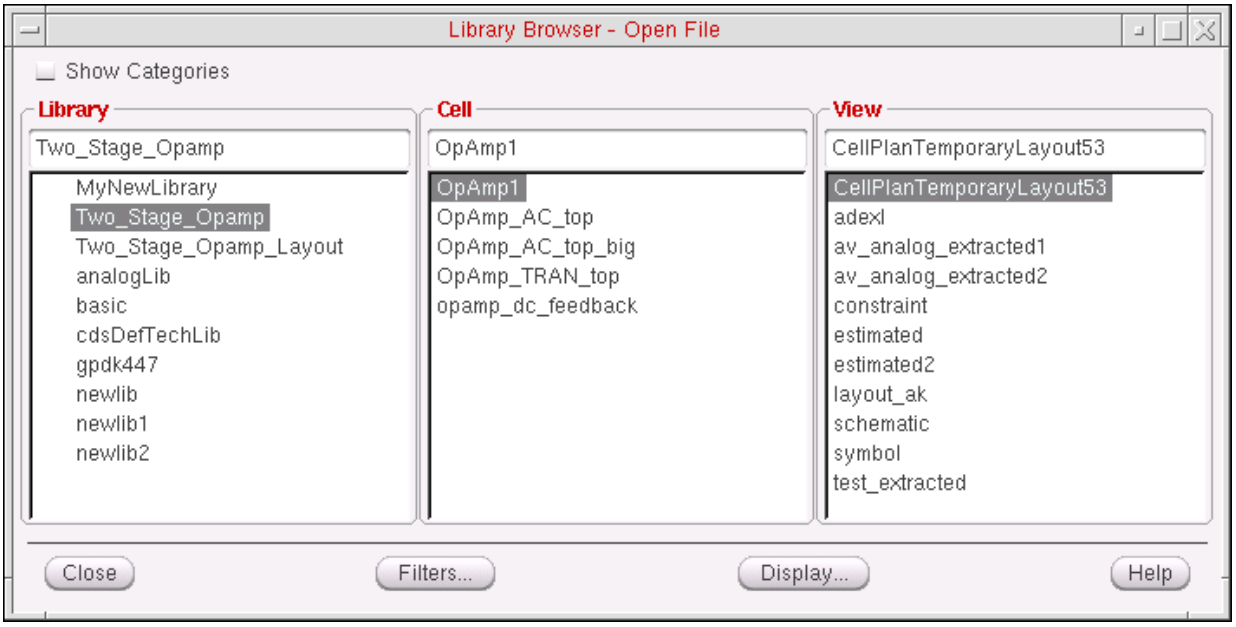
Once the cell name is entered in the *Cell* field, the views available for that cell will appear in the *View* drop-down combo box.

c. In the *View* drop-down combo box, select a view name.

Note: If a thumbnail of the selected cellview is available, this will be displayed in the Open File form as an aid to selection. If no thumbnail is available, the space will be blank (for more information on thumbnails, see Thumbnail Images of Cellviews in the *Cadence Library Manager User Guide*.

Tip

Alternatively, you can click *Browse* to open the *Library Browser – Open File* window and select your design.



2. In the *Application* group box, select an item from the *Open with* drop-down combo box. For example:

For schematic cellviews:	*ADE Assembler* *ADE Explorer* *SMG Model* *Schematic* *Schematics L* *Schematics XL*	The choices you can make depend on the application levels you have licensed at your site. See also "Using Different Application Tiers" on page 62.
For symbols	*Symbol L* *Symbol XL*	
For layout views:	*Layout ViewerLayout XL* *Layout EXL*	
For config views:	*Hierarchy Editor*	This is the only choice.
For an ADE state view:	*ADE State*	This is the only choice.
For text views:	*Text Editor*	This is the only choice.

3. (Optional) Mark the *Always use this application for this type of file* check box if you want the program to use the selected application when opening a view that is the same file type as what you specified in the *View* field.

4. Select one of the *Open for* radio buttons: *edit* or *read*.

To open the file in edit mode, you must have write permission to the file.

See als Setting Default Cellview Open Mode.

5. (Optional) If you chose *File – Open* in a session window, you can select one of the *Open in* radio buttons to indicate how you want the program to open the cellview:

- *new tab* opens the cellview on a new tab in your current session window.
- *current tab* opens the cellview on the current tab in your current session window.
- *new window* opens the cellview in a new session window.

6. Click *OK*.

The cellview appears in the specified application design window according to your selections. If the appropriate application design window is not already open, the software starts the application.

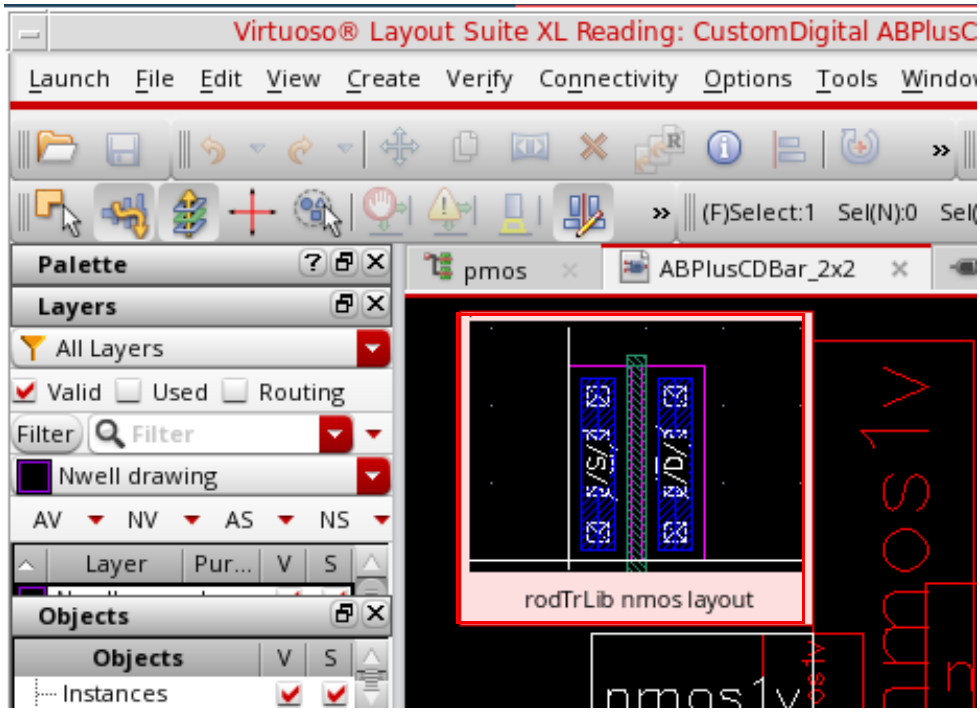
You can use the following variable to set the stoplevel of the thumbnail in the cellview window.

```
ui.thumbnails stopLevel 'int -1
```

If you set any value, which is >=0, then that value will be used instead of the system default value, 0 during thumbnailing.

Enabling Design Preview

If multiple designs are open, place the mouse pointer on a tab to preview the design open in that tab, as shown below:



To enable this feature, you need to add the following variable in the `.cdsenv` file:

```
ui  imageTabTip boolean t
```

Setting Default Cellview Open Mode

You can specify the default mode, *edit* or *read*, for opening cellviews in the Open File form using the `deOpenFormAccessModeAlwaysRead` environment variable in your `.cdsenv` file.

For example:

```
designEditor.fileSpec deOpenFormAccessModeAlwaysRead boolean nil nil
```

Here, if `deOpenFormAccessModeAlwaysRead` is set to `t`, the *Open for* option, in the Open File form, will always be set to *read* irrespective if a cellview is writable or not.

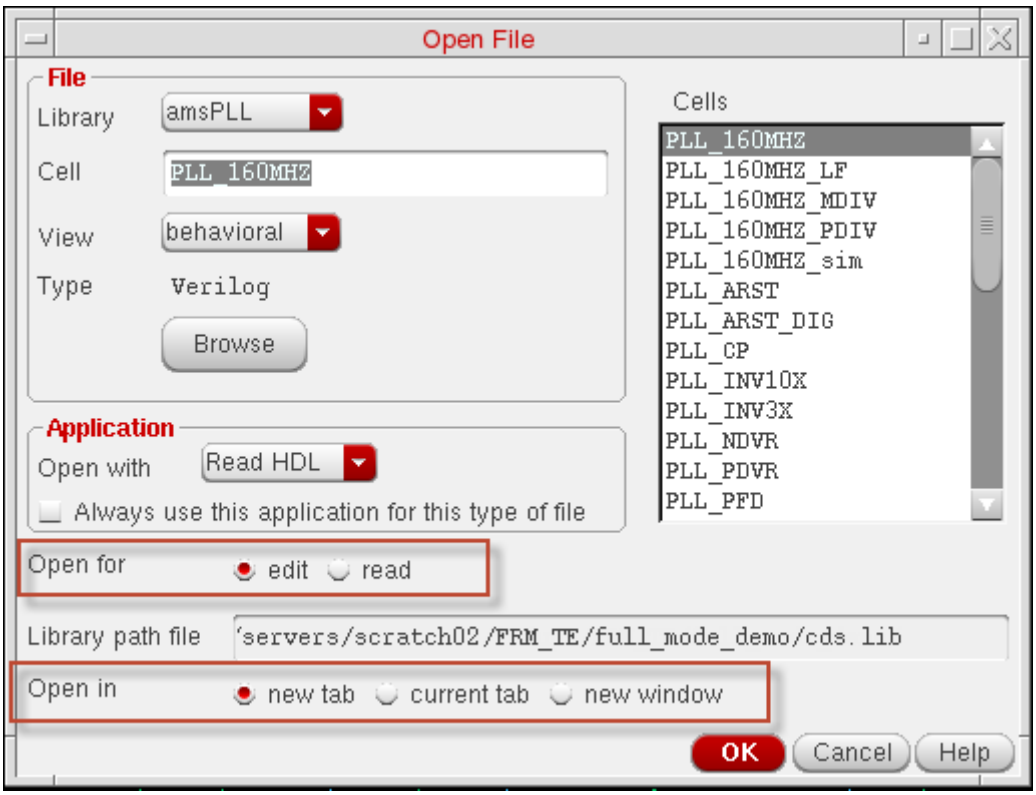
Note: This environment variable setting will only impact the default setting for the *Open for* option in the Open File form.

You can set the following environment variable in your `.cdsenv` file to define your preference in opening a cellview in a new tab, the current tab, or a new window. The default option is a new tab.

```
designEditor.fileSpec deDefaultWindowStyle
```

The Open File form, accessed from the *File — Open* option of Cadence applications like Schematic Editor, displays the *Open In* options based on how you have set `designEditor.fileSpec` in your `.cdsenv` file.

The following figure illustrates in the *Open for* and *Open in* options on the Open File form.



Controlling the Fit-on-Open Feature

You can specify the number of attempts made to open a window at a certain zoom level for a graphics Window in Virtuoso. Valid values are any integer number greater than or equal `0`. You can set this number based on your working environment.

`designEditor.window fitViewAttempts int` *`number_of_attempts`*

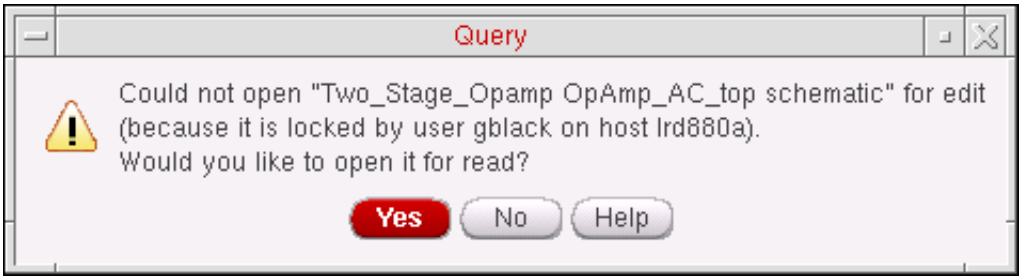
Setting the value to `0` (zero) disables the fit-on-open feature.

Default value: `1`

Cellview Already in Use

If you attempt to open a cellview that is already being used (that is, the cellview file is locked by another user), a warning message will be displayed informing you of the current user of the cellview and the current host machine name.

From here you can choose to either open the cellview as read-only or not to open the cellview.



Note: This warning message will also be displayed if you attempt to descend into, or edit in place, a cellview currently in use by another user.

Indication of Modified Cellviews

Design Editor title bar indicates if a cellview has been modified, but not saved. It uses asterisk (*) as the default indicator at the end of the cellview name in the title bar to indicate a modified cellview.

You can change the default indicator using the environment variable `cellviewModifiedIndicator`. Design Editor reads this environment variable once and uses the indicator value throughout a DFII session. If the environment variable is modified during a session, a warning message is issued once. For details, see "Customizing Titles for Windows and Icons" on page 204.

Cellview History

Virtuoso Design Environment maintains the history of cellviews you access. Using this history, the environment displays a list of cellviews you access in the CIW *File* menu. You can open a recently accessed cellview directly from the *File* menu.

Virtuoso Design Environment does not consider the application tier that was used to open the cellview previously. For example, open a cellview in Layout EXL and close it. The CIW *File* menu should now include an entry to access this cellview. Open the cellview from the CIW *File* menu. The cellview opens in Layout XL and not Layout EXL.

Using the environment variable `ignoreAppTierInHistory`, you can control whether Virtuoso Design Environment must consider the application tier recorded in the history for opening a cellview. If you want to ensure the application tier is considered, set this environment variable in your `.cdsenv` file as `nil`. By default, this variable is set to `t`. For example:

```
designEditor.history ignoreAppTierInHistory boolean t
```

This means that application tier saved in history is ignored when an item is selected from the history menu. In ICADVM20.1, Layout XL is used to open a layout cellview by default.

Filtering Unavailable Tiers when Opening a Cellview

When a cellview is saved in Virtuoso, information about its contents is saved in the property bag, which is the `data.dm` file. When the cellview is reopened from the Open File form, information in the property bag is read to determine the application tiers that support the data contained in the cellview. Based on this information, only supported tiers are displayed in the Open File form.

For example, consider that you have a layout cellview and you create a virtual hierarchy in it. You need Layout EXL to open this cellview because it is not supported in Layout XL. The information that cellview uses virtual hierarchy is saved in its `data.dm` file. So, next time when you open this cellview, you will see that only Layout Viewer and Layout EXL are the available choices to open the cellview and Layout XL is not available.

Setting the Default Application for a Cellview

You can set the default application for a schematic, symbol, or layout cellview. See

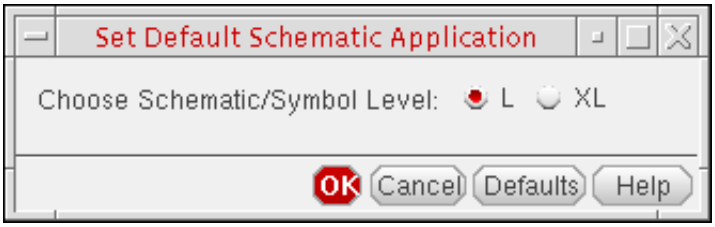
- Setting the Default Application for a Schematic or Symbol Cellview on page 112
- Setting the Default Application for a Layout Cellview on page 113

Setting the Default Application for a Schematic or Symbol Cellview

To set the default application for a schematic or symbol cellview, do the following:

1. In the session window, choose *File – Set Default Application*.

   The Set Default Schematic Application form appears.



2. For *Choose Schematic/Symbol Level*, select one of the following application levels:

   - *L*

     Note: You can specify this setting in your `.cdsenv` file as follows:

     ```
     graphic schematicDefaultTier string "L"
     ```

   - *XL*

     Note: You can specify this setting in your `.cdsenv` file as follows:

     ```
     graphic schematicDefaultTier string "XL"
     ```

3. Click *OK*.

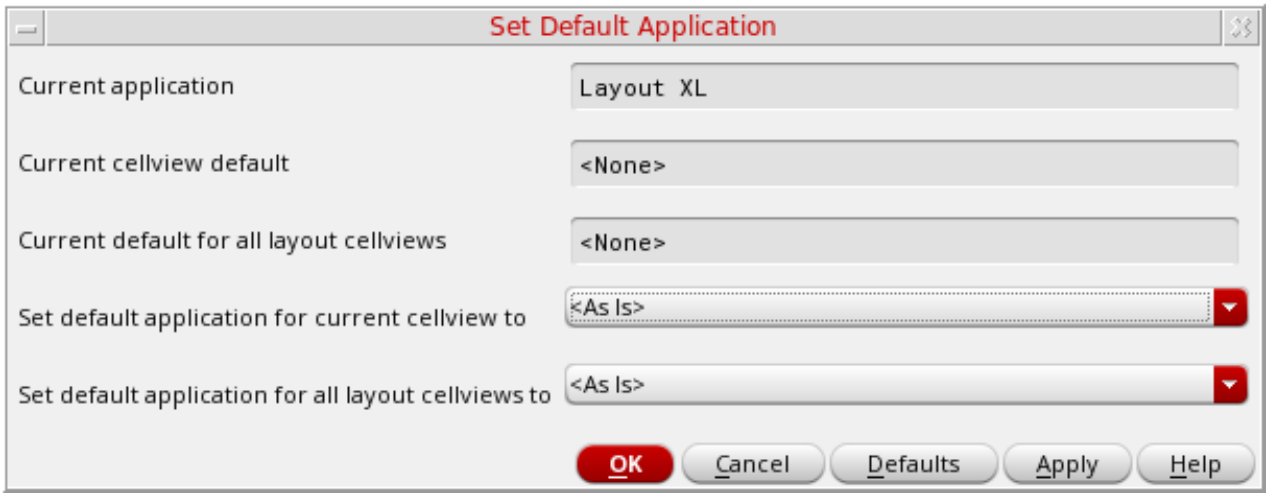
   The program uses the application level you selected the next time you open a cellview and writes the environment setting to your `.cdsenv` file.

Setting the Default Application for a Layout Cellview

To set the default application for a layout cellview, do the following:

1. In the session window, choose *File – Set Default Application*.

   The Set Default Application form appears.



2. In the *Set default application for current cellview to* drop-down combo box, select a default application to use when opening the current cellview.
3. (Optional) In the *Set default application for all layout cellviews to* drop-down combo box, select a default application to use when opening any schematic cellview.

   Note: For a specific cellview, the *Set default application for current cellview to* setting overrides the *Set default application for all layout cellviews to* setting.
4. Click *OK*.

   The program uses the selected application when opening the designated cellview type.

Closing a Cellview on a Tab in a Session Window

To close a cellview on a tab in a session window that has more than one tab, do the following:

1. Right-click the tab you want to close.

   A pop-up menu appears.

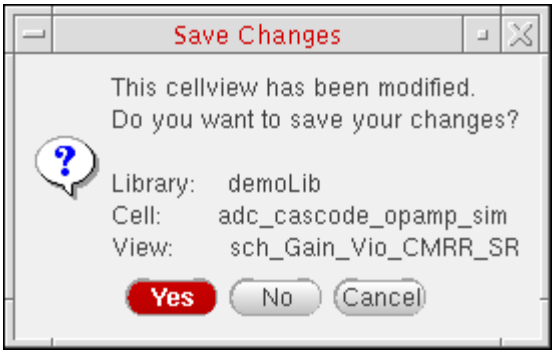
2. Choose *Close Tab*.

   The tab closes.

Alternatively, do the following:

1. Select the tab you want to close.
2. Do one of the following:
   - Choose *File – Close*.
   - Choose *Window – Tabs – Close Current Tab*.
   - Click the **X** button on the right side of the tab. 

   If you have any unsaved changes for that cellview, the Save Changes dialog box is displayed.



   - To save changes, click *Yes*.
   - To discard changes, click *No*.

   The tab closes.

   Note: If you click *Cancel* on the Save Changes prompt, the program does not save changes and does not close the tab.

Closing All but the Cellview on the Current Tab

To close all cellviews except the one on the current tab (in a session window that has more than one tab), do the following:

1. Right-click the tab for the cellview you want to keep open.

   A pop-up menu appears.

2. Choose *Close Other Tabs*.

   Cellviews on all other tabs are closed leaving only the current cellview open in your session window.

Alternatively, do the following:

1. Select the tab for the cellview you want to keep open.
2. Choose *Window – Tabs – Close Other Tabs*.

   Cellviews on all other tabs are closed leaving only the current cellview open in your session window.

Copying the Current Cellview to a New Session Window

To copy the current cellview on the current tab to a new session window, do the following:

➤ Choose *Window – Copy Window*.

The current cellview on the current tab appears in a new session window. The program preserves your application level and workspace configuration (same set of assistant panes) in the new session window.

However, the *Copy Window* option is disabled in case the Pcell IDE assistant is opened in the layout window. This is because only one Pcell IDE session is supported at a time.

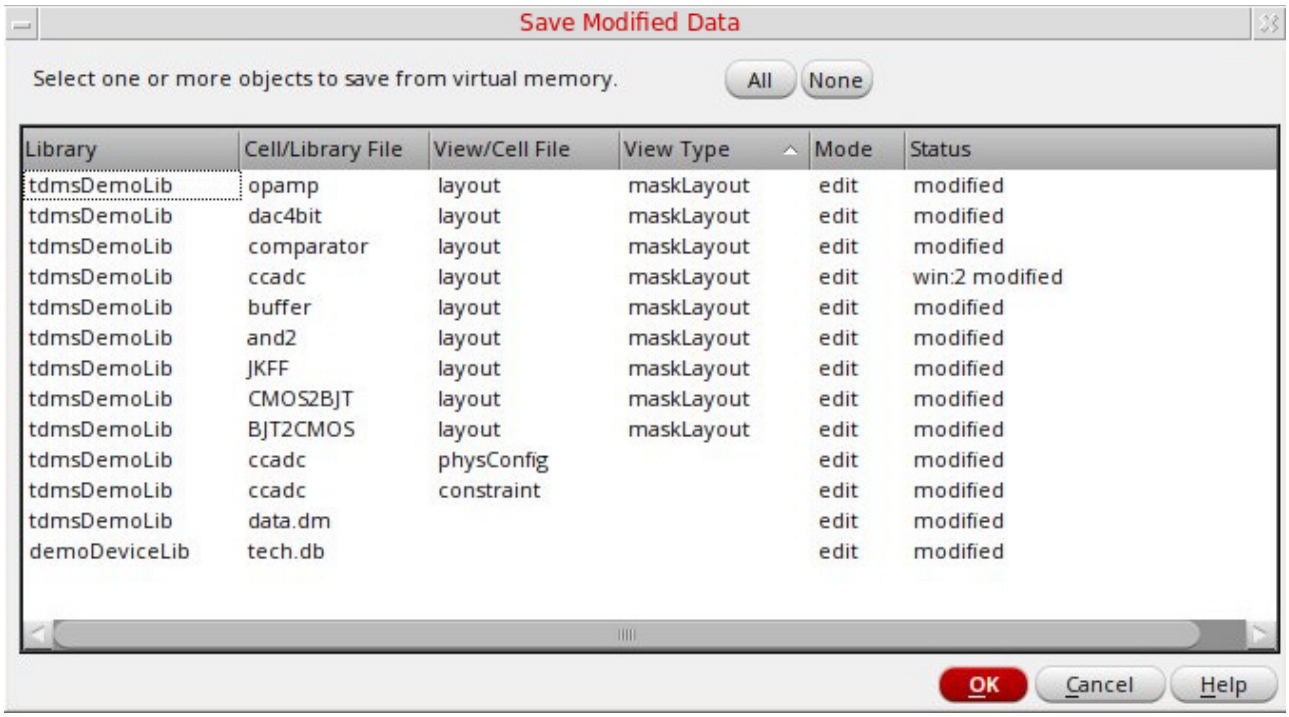
Saving Modified Data

You can save all currently open and modified cellviews and data using the Save Modified Data form. The form checks for five types of data - cellviews, physical configuration, constraints, property bag (`data.dm`), and technology files (`tech.db`). The form lets you save this data from the virtual memory.

To save all open and modified cellviews:

1. In the CIW, choose *File – Save Data*.

   The Save Modified Data form appears.



Tip

You can sort the information by clicking any of the column headers.

2. Specify the data to save as follows:

   - Keep the `Ctrl` key pressed and click discrete rows in the table.
   - Keep the `Shift` key pressed and click two rows to select them and all the rows in between.
   - Click *All* to select all rows.

- Click *None* to deselect all rows and start selecting again.

3. Click *OK*.

All modified cellviews are saved but stay open for any further editing.

Related links

Save Modified Data

ddsHiSaveData

Closing All Cellviews

You can close all open data using the Close and Purge Data form. The form checks for five types of data - cellviews, physical configuration, constraints, property bag (`data.dm`), and technology files (`tech.db`).

This form can display data that have lost connection to their physical files due to one of these reasons:

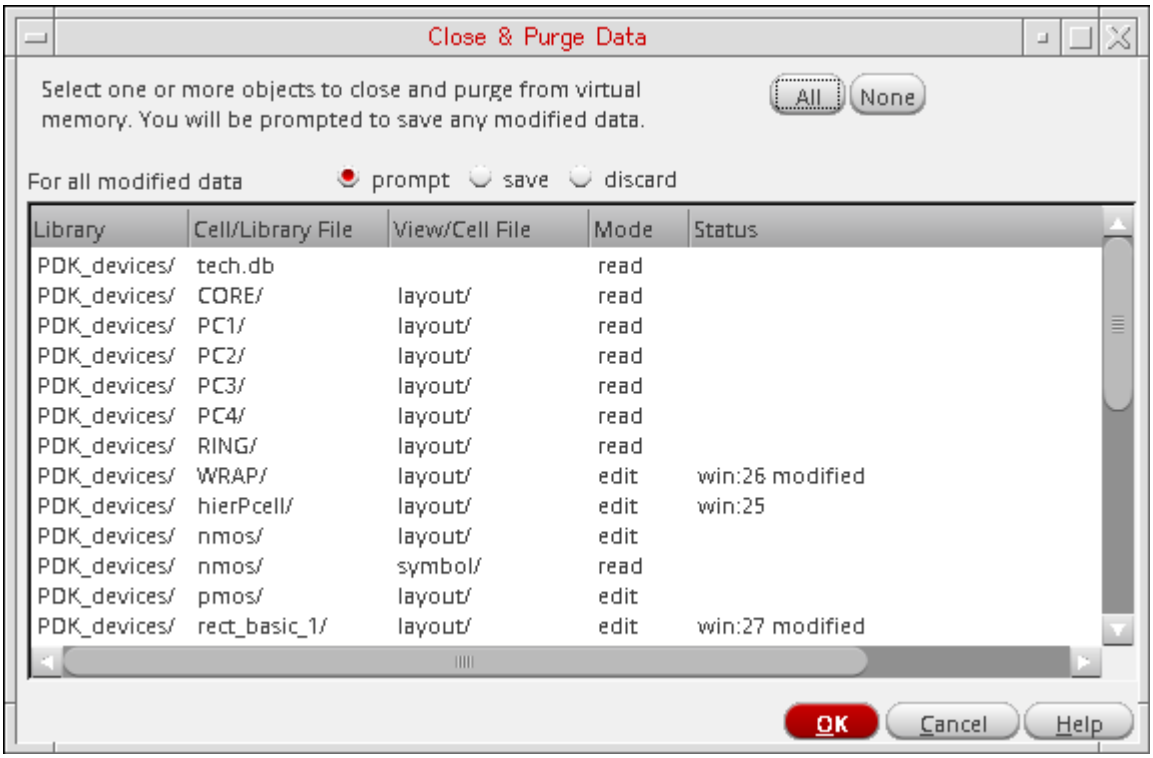
- A network failure, due to which data on the shared file servers becomes inaccessible
- Deletion of local data
- Renaming of local data

The form lets you purge such data from the virtual memory. It displays relevant warnings in CIW.

Instead of closing each cellview individually, you can close all cellviews at once as follows:

1. In the CIW, choose *File – Close Data*.

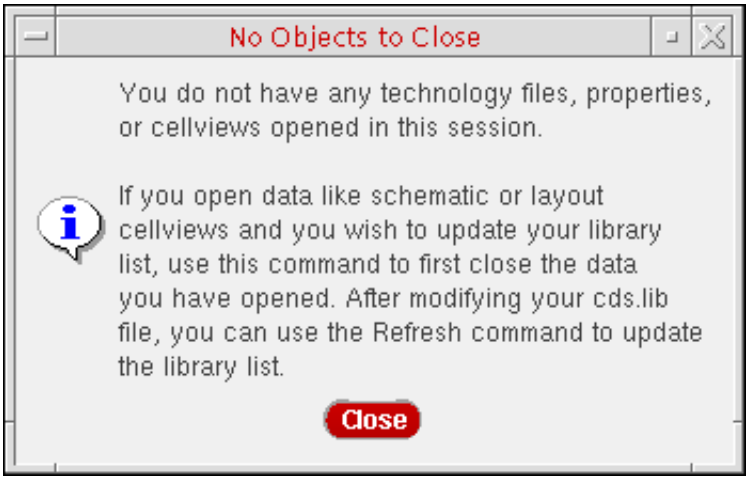
   The Close and Purge Data Form appears.



Tip

You can sort the information in the Close and Purge Data form by clicking the column header of interest.

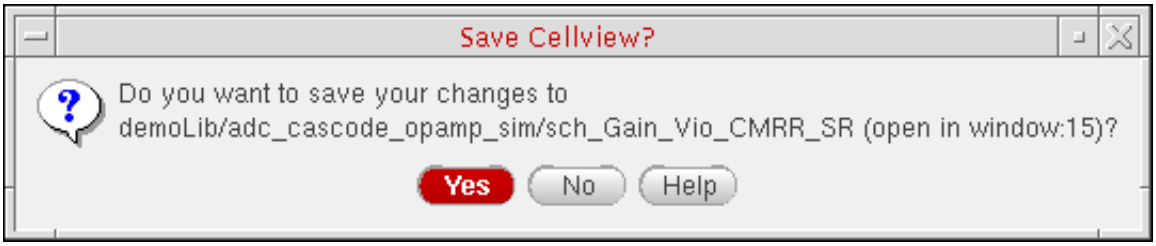
If you do not have any cellviews open, the No Objects to Close prompt appears.



1. Choose how Virtuoso handles modified data on *File – Close Data* selection:
   - *prompt*: always prompt to save data on close.
   - *save*: automatically save data on close.
   - *discard*: do not save data if Virtuoso closed using *File – Close Data*.
2. On the Close and Purge Data form, do one of the following:
   - Click *All* to select all cellviews listed on the form.
   - Click *None* to deselect all cellviews (and thus not close any).
   - Click and drag to select a contiguous set of cellviews.
   - `Control`-click individual (noncontiguous) cellviews to select each one.

3. Click *OK*.

   If you have modified a cellview, the Save Cellview? prompt appears.



   - To save the edits before closing and purging, click *Yes*.
   - To discard the edits before closing and purging, click *No*.

   The cellviews you selected are closed and their data purged from virtual memory.

Specifying the Default Instance Prefix

When you place an instance, the default alphabetic character prefix is `I` for instances and `M` for mosaic instances (an array of instances). You can change these default values by placing a property on the master cellview or setting a variable in your `.cdsenv` file.

■ Changing the Prefix Using a Property on page 121

■ Changing the Prefix Using an Environment Setting on page 122

If you do not specify an instance name explicitly, the program checks the master cellview for the presence of the appropriate property as follows:

■ For an instance: `instNamePrefix`

■ For an array of instances: `arrayInstNamePrefix`

When you place the instance, the program does the following:

1. Uses the prefix specified by the appropriate master cellview property, if it exists. Otherwise,
2. Uses the prefix specified by the `cdba` variable in your `.cdsenv` file. For example:

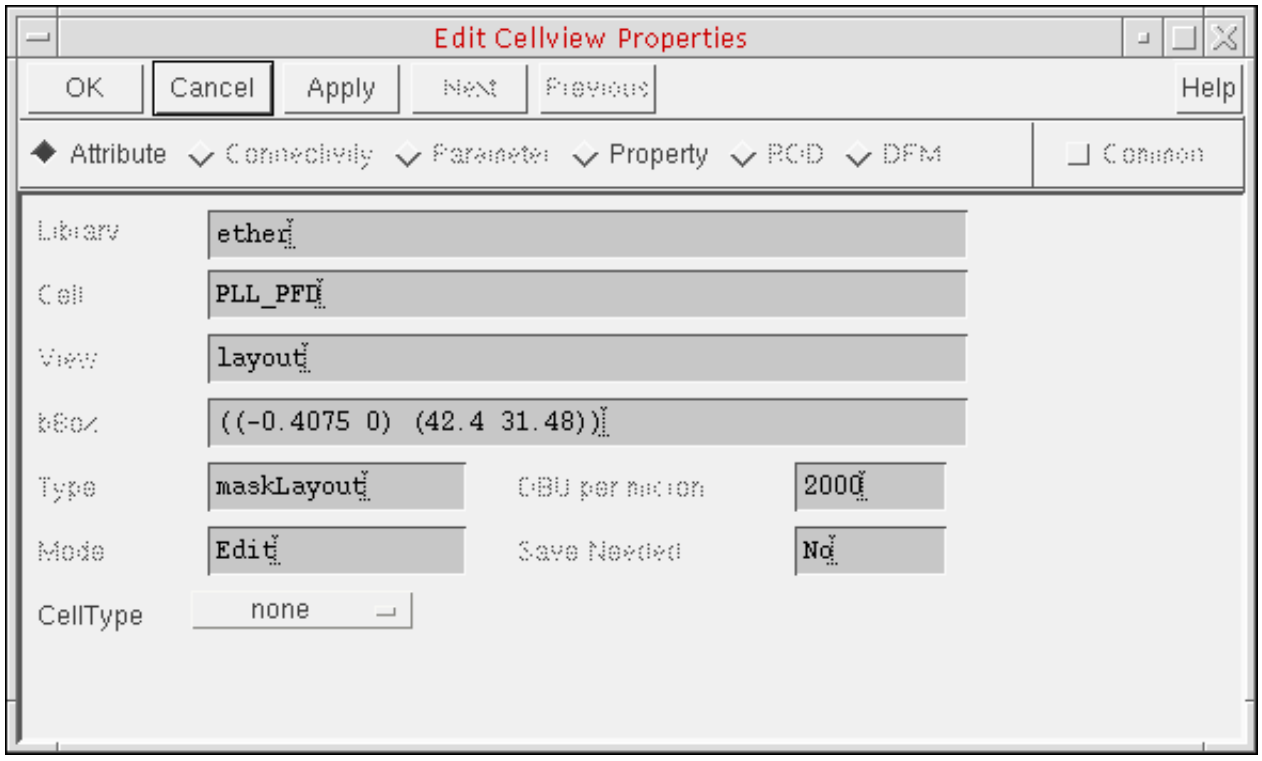
```
cdba    dbInstNamePrefix        string   "I"
cdba    dbArrayInstNamePrefix   string   "M"
```

Changing the Prefix Using a Property

To change the default prefix by adding a property to the master cellview, do the following:

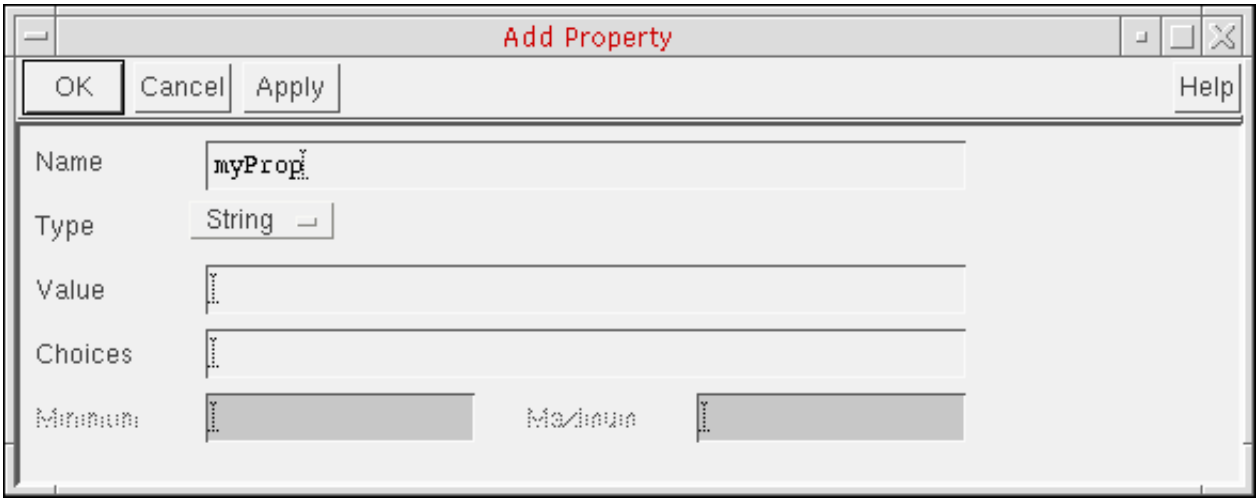
1. Open the desired layout master cellview.
2. Select *File – Properties*.

   The Edit Cellview Properties form appears.



3. Select *Property*.
4. Click *Add*.

The Add Property form appears.



String appears as the default *Type*.

5. In the *Name* field, type the property name:
   - `instNamePrefix` for scalar instances
   - `arrayInstNamePrefix` for an array of instances (simple mosaics)
6. In the *Value* field, type the prefix you want the program to use when you place an instance.
7. Click *OK*.

   The property name and value appear on the Edit Cellview Properties form.
8. Click *OK*.

   The program applies the property to the master cellview.

Changing the Prefix Using an Environment Setting

To change the default prefix using an environment setting, do the following:

1. Copy the default `.cdsenv` file from the `samples` directory (see "Copying and Editing the Default .cdsenv File" on page 187).
2. Change the following line or lines (depending on which prefix or prefixes you want to change) to your custom `.cdsenv` file:

```
cdba    dbInstNamePrefix        string   "yourCustomInstNamePrefix"
cdba    dbArrayInstNamePrefix   string   "yourCustomArrayInstNamePrefix"
cdba    dbAddCellNameToInstNamePrefix   boolean   t
```

The `dbAddCellNameToInstNamePrefix` environment variable enables you to append the cell name to the instance name prefix. The default value is `nil`. For example, your instance name prefix is `INS` and if you set this environment variable to `true`, it will generate `INS_<cellname>_<number>`.

Here, the <*cellname*> is the cell name and <*number*> is a uniquely generated number.

These settings apply the next time you start the software.

6

Customizing Your Design Environment

You can use the *Options* menu in your Command Interpreter Window (CIW) to customize your Virtuoso Design Environment:

Note: You can customize specific application setup information by referring to the specific application documentation (for example Customizing the Virtuoso Schematic Editor and Customizing the Layout XL Desktop). Customization updates made via the CIW are common to multiple users or global to an entire site.

- Specify user preferences such as: window placement, configuration, and behavior; command input behavior, mouse behavior, and web browser; CIW configuration (see "Specifying User Preferences" on page 127)
- Specify when the Library Browser form appears and whether the Exit prompt appears (see "Specifying , Library Browser, and CIW Preferences" on page 141)
- Save and restore window position between sessions (see "Saving and Restoring Window Positions" on page 146)
- Specify what information appears in the output area of the Command Interpreter Window (see "Changing the Log Filter Options" on page 148)
- Check licenses in and out and specify automatic check-in and check-out behavior (see "License Management Using the Software Product License Management Form" on page 60; see also "Controlling Automatic Check-In Behavior" and "Controlling Automatic Check-Out Behavior" in the *Cadence Library Manager User Guide*)
- Define banner menus for the CIW and other licensed applications (see "Customizing the Menu Banner" on page 150)
- Customize toolbars (see "Customizing Toolbars" on page 151)
- Assign bindkeys to keyboard keys and mouse clicks (see Appendix A, "Bindkeys and Access Keys")
- Specify the default prefix for instances (see "Specifying the Default Instance Prefix" on page 120)

See also

- Understanding the .cadence Hierarchy on page 170
- Chapter 8, “Saving and Recalling Default Settings”

You can customize your design environment such that your settings are loaded for all subsequent sessions by editing various dot files in your home directory (such as `.cshrc`, `.cdsinit`, and `.Xinitrc`). Your settings remain in effect until you edit the dot files again. See also Chapter 7, “Specifying Environment Settings.”

Note: The `.cshrc` and `.Xinitrc` files are system specific files with content both relevant and irrelevant to Cadence and Virtuoso applications. The `.cdsinit` file applies solely to Cadence and Virtuoso software.

To customize this	Edit this file
UNIX paths, file locations, and device specifications	`.cshrc`, `.login`, or `.profile` file
General window appearance and behavior	`.Xdefaults`, `.xsession` or desktop/window manager configuration
Window appearance and command actions and environment variables for Cadence software	`.cdsinit` or `.cdsenv` file

Specifying User Preferences

You can specify user preferences for the current session using the User Preferences form. To open the User Preferences form, do the following:

➤ From the CIW, choose *Options – User Preferences*.

The User Preferences form appears.

Changes you make on this form, once applied, take effect with the next command you issue or the next window you open. *Menu Shortcuts* is an exception since the changes made to it will only take place in the next Virtuoso session if you permanently save the changes in the `.cdsenv` file.

Tip

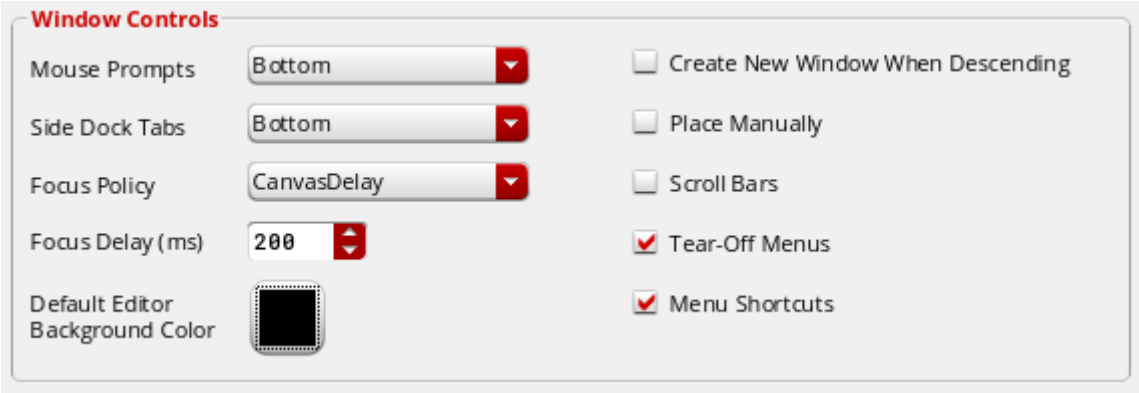
To make these changes permanent, choose *Options – Save Defaults* (see "Specifying New Default Values" on page 216) or edit your `.cdsenv` file. Some of the changes will become effective with the next command while others become effective the next time a window is opened.

The following procedures are presented:

- Specifying Window Controls on page 128
- Specifying Command Controls on page 134
- Specifying CIW Controls on page 136
- Specifying Dashboard Controls on page 140

Specifying Window Controls

Specify window controls, which are settings that affect the position and appearance of windows in the Virtuoso user interface, by using the following options on the User Preferences form (in the CIW, choose *Options – User Preferences*):



1. In the *Window Controls* group box, check or clear the following check boxes to indicate the settings you want:

Check Box	Description
Place Manually	Clicked: A placement cursor when a new window opens so that you can click and drag the size and location of the window. Cleared: New windows open using a default size and location.
Create New Window When Descending	Clicked: A new window opens when you choose *Design – Hierarchy – Descend*… in the Schematic Editing window. Cleared: The new view replaces the old view in the same window when you choose *Design – Hierarchy – Descend*… in the Schematic Editing window. **Note:** The above will depend on tab settings and choices.
Scroll Bars	Clicked: Scroll bars appear along the right side and bottom of a new window whenever the design view does not fit entirely in the viewing area of the window. Cleared: Scroll bars do not appear.
Tear-Off Menus	Clicked: Future menus have tear-off handles. Cleared: Future menus do not have tear-off handles.
Menu Shortcuts	Clicked: You can type *Alt* together with the underlined character for a menu (such as the *F* for the menu in the CIW) to access that menu, and you can type the underlined character on a menu item (such as the *O* in *Open* on the menu) to access that item. **Note:** Changing this option will have no effect during the current session. Cleared: You cannot access menus or menu items using keyboard shortcuts.

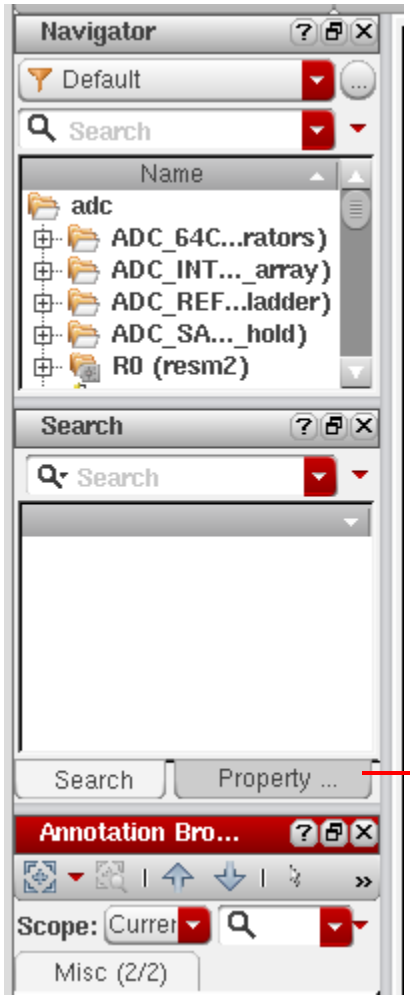
2. (Optional) From the *Mouse Prompts* drop-down list box, select one of the following:
   - *Top* if you want the prompts to appear along the top edge of each new window
   - *Bottom* if you want the prompts to appear along the bottom edge of each new window
   - *None* if you do not want the prompts to appear at all

     Note: The *Mouse Prompts* preferences are overridden by any applied workspace regardless of the selected position. Also, if the mouse prompts are set to *None* before the window is created, then the mouse prompts (containing the mouse bindings line) do not appear even if a workspace has the designated mouse prompts location.
3. (Optional) From the *Focus Policy* drop-down list box, select one of the following:
   - *Click* if you want to click in the session window or the dockable window to be in focus
   - *CanvasDelay* if you want a session window canvas to be in focus if the pointer remains in it for a specific period of time. This is also the default focus policy.
   - *CanvasAsstDelay* if you want both the docked assistant windows or the canvas to get focus when the pointer remains in either for a specific period of time

   Note: In Virtuoso, the focus policy applies only to docked assistants and does not apply to floating dockable windows since that is controlled by window manager options.
4. Change the *Focus Delay* slider to set the specific period of time for which the pointer must remain in a session window canvas to get focus. The default value is 200 milliseconds, which is 1/5 of a second.
5. (Optional) Use the *Default Editor Background Color* option to change the background color of the editor window. On clicking this option, the Select Color window displays, from where you can choose the required color.

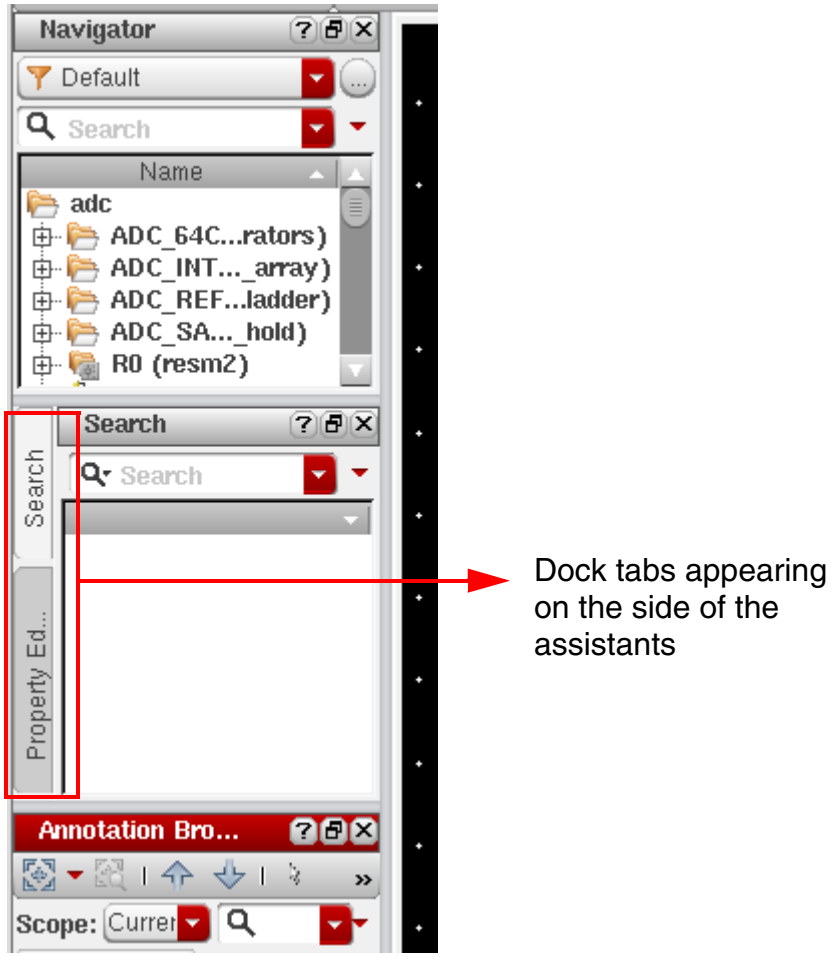
   Default: `Black`
6. In the *Side Dock Tabs* drop-down list box, select one of the following:

- *Bottom* if you want the tabs for multiple assistants, which are open at the same time, to appear at the bottom of a session window. This is shown in the following figure:



Dock tabs appearing at the bottom of the assistants

- *Side* if you want the tabs for multiple assistants, which are open at the same time, to appear on the side, as shown in the following figure below:



For more information on the `sideDockTabs` environment variable, see Specifying Window Controls in .cdsenv.

1. Click *OK* (or *Apply* if you do not want to dismiss the form).

   The program applies your settings to subsequently opened windows.

Save your settings and apply them to future sessions through the following steps:

1. In the Command Interpreter Window (CIW), choose *Options – Save Defaults*.

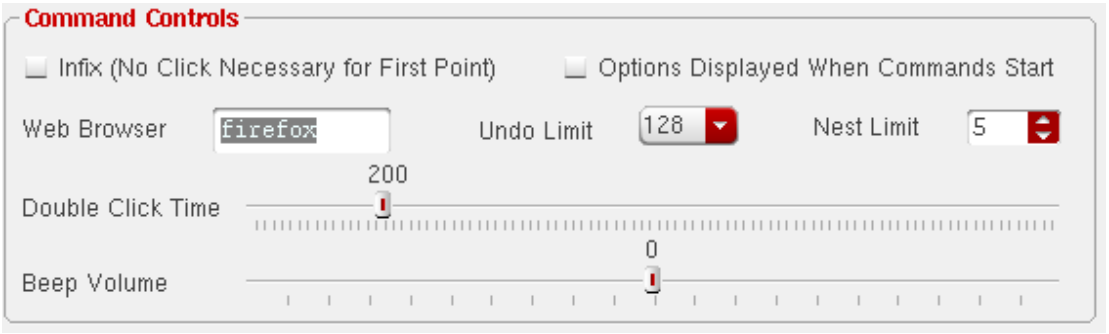
   The Save Defaults Form appears.

2. Click *OK* to close the form (or *Apply* if you do not want to dismiss the form).

The program saves these settings to your `.cdsenv` file and applies them to future sessions (see Specifying New Default Values).

Specifying Command Controls

Specify command controls, which are settings that affect the behavior of commands and the mouse by using the following options on the User Preferences Form (in the CIW, choose *Options – User Preferences*):



1. In the *Command Controls* group box, mark or unmark the following check boxes to indicate the settings you want:

Check Box	Description
Infix (No Click is necessary for first point)	If you mark this check box, the current location of the cursor in a design window is taken as the first point for the command (the enterfunction) when you enter the bindkey for the command. If you do not check this check box, you must click the mouse to enter the first point after invoking the command.
Options Displayed When Commands Start	If you mark this check box, the options form associated with an enterfunction appears when you enter the bindkey for the enterfunction. If you do not mark this check box, you must use the F3 key, or whatever key is bound to the `hiToggleEnterForm` command, to open the options form. (If the options form is open, the `hiToggleEnterForm` command closes it.) **Note:** Not all enterfunctions have an options form. This setting does not apply to enterfunctions that do not have one.

2. (Optional) In the *Web Browser* field, type a browser executable (default is `firefox`).

   Note: The directory that contains the browser executable must be in your path.

3. (Optional) In the *Undo Limit* field, select one of the following states:

Undo Limit	Description
128	Turns on the Undo feature, with a limit of 128 undos
0	Turns off the Undo feature

   Note: The above settings are related to database (graphic) operations and do not impact individual application settings.

4. (Optional) In the *Nest Limit* field, type or select a different maximum level of nesting.

5. (Optional) Change the *Double Click Time* by moving the slider to the left or right.

   Double Click Time is given in milliseconds. 200 ms is 1/5th of a second. If the program receives a second mouse click within the time period defined by *Double Click Time*, the action is taken as a double-click.

6. (Optional) Change the *Beep Volume* by moving the slider to the left or right.

   Note: *0* means the normal/default volume level for your system.

   - A move to the left decreases the beep volume.

     -100 turns the beep sound off.

   - A move to the right increases the beep volume.

     100 is the loudest volume possible.

     Note: For systems that do not support volume control, any setting from *-99* to *100* turns on the beep sound.

7. Click *OK* to close the form (or *Apply* if you do not want to dismiss the form).

   The User Preferences form closes. Your settings are applied to this session only.

Save your settings and apply them to future sessions by doing the following:

1. In the Command Interpreter Window (CIW), choose *Options – Save Defaults*.

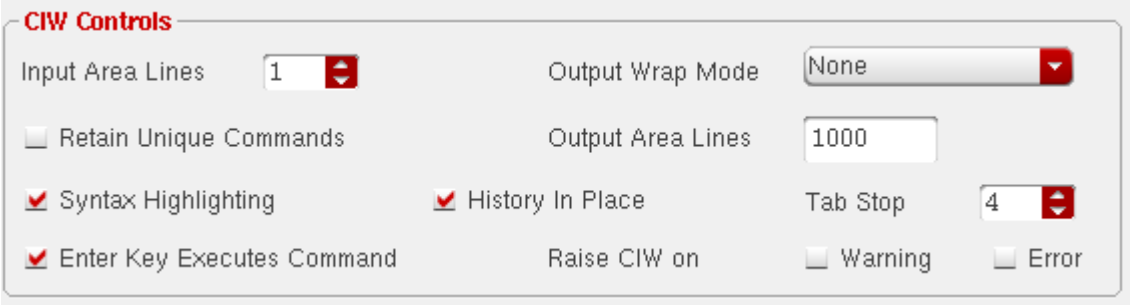
   The Save Defaults form appears.

2. Click *OK*.

The program saves these settings to your `.cdsenv` file and applies them to future sessions (see "Specifying New Default Values" on page 216).

Specifying CIW Controls

Specify settings that affect the appearance and configuration of the Command Interpreter Window (CIW) by using the following options on the User Preferences form (in the CIW, choose *Options – User Preferences*):



1. Select a number in the *Input Area Lines* spin box to specify the number of previous commands that appear in the input area.

   If you specify more lines than are available in the window, the program reduces the size of the output history area to allocate more space to the input area.

   Additionally, the height of the CIW will also be modified when the number of input area lines is increased to the point where the CIW is not of a sufficient size to accommodate the changed setting.

   You can also use the `ciwCmdInputLines` environment variable to amend the input area line settings (for more information see Specifying CIW Controls in .cdsenv).

2. From the *Output Wrap Mode* drop-down list box, select one of the following values to wrap the messages in the CIW:

   - *None* if you do not want to wrap any messages that appear in the CIW. This is also the default value.
   - *Word* if you want to wrap a message between words. This means that words longer than the width of CIW will appear in the next line and you will see a horizontal scroll bar
   - *Anywhere* if you want to wrap a message between characters. This means that words longer than the width of CIW will appear broken and you will never see a horizontal scroll bar

- *Word or Anywhere* if you prefer wrapping between words but in case a word is wider than the CIW, messages will be wrapped between characters. Therefore, in this case too you will never see a horizontal scroll bar.

3. Select the *Retain Unique Commands* check box to remove identical commands from the input area of the CIW. By default, this check box is not checked.

   For more information, see "Repeating a Command" in the "Using the Command Interpreter Window" chapter.

4. Use the *Output Area Lines* field to specify the maximum number of lines contained in the output area (accessible by scrolling when the number of lines exceeds the height of the output area).

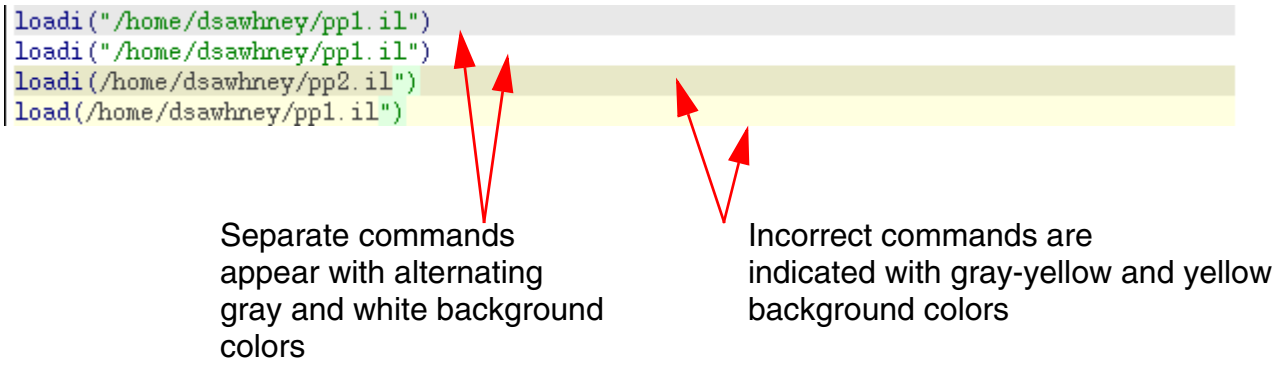
   Note: If you specify a number that is smaller than the current setting, the program truncates the number of lines contained in the output area.The User Preferences form closes. Your settings are applied only to this session.

Important

The minimum valid value for this setting is 100.

5. Select the *Syntax Highlighting* check box to enable checking the syntax or commands entered in the input area of the CIW. By default, this check box appears selected.

   Separate commands appear with alternating white and gray background colors. Incorrect or incomplete commands are indicated with light yellow or light gray-yellow background colors, respectively. These are shown in the following figure:



If syntax highlighting is turned off, that is, the *Syntax Highlighting* check box is not selected, all the commands that you enter in the input area will still appear with alternating background colors but no syntax checking will be done.

For more information, see Typing SKILL Commands in the "Using the Command Interpreter Window" chapter.

6. Select the *History In Place* check box if you want to access history commands within the "edit area" of the input area (the edit area is the bottom portion of the input area below the last displayed history line) using the up and down arrow keys. If the check box is not selected, the up arrow key will traverse into the displayed history buffer within the input area. If the check box is selected, you can still traverse into the displayed history buffer by using the left arrow key or by clicking into the area above the edit area.

   For more information, see Repeating Commands from CIW in the "Using the Command Interpreter Window" chapter.

   Note: The CIW built-in history is different from using the `history()` command. The built-in history only displays typed-in commands, whereas, the `history()` command also lists "accelerated" commands (those commands that were run by selecting menus or using bindkeys). For information about the history command, see `history()` in the *Ocean Reference* user guide.

7. Select a number for the *Tab Stop* spin box to specify the number of character spaces that will represent a tab in both, the input and output areas of the CIW.

8. (Optional) Select or clear the *Enter Key Executes Command* check box to indicate the setting you want.

   - When selected, you can press *Enter* to execute a command block typed in the input area.

     The command block that contains the cursor will be executed regardless of the position of the cursor within that block. In the following example, ^ marks the position of the cursor when you press *Enter*:

     ```
     foreach(term dbGetNetParent printf("%s\n" term->name()))
                                 ^
     ```

     The program evaluates and executes the entire command block.

     When the *Syntax Highlighting* check box is selected, use any of the following key combinations: `Shift+Return`, `Shift+Enter`, `Ctrl+J`, or `Ctrl+M` to enter a newline when the *Enter Key Executes Command* check box appears selected.

     When the *Syntax Highlighting* check box is not selected, all of the key combinations mentioned above will execute the command block, that is, send the command block to the SKILL parser. This is because if the commands are not complete, they will not be executed by the SKILL parser until SKILL receives the remainder of the commands.

   - When clear, pressing the *Enter* key will insert a newline. To execute a command when the *Enter Key Executes Command* check box is not selected, use any of the following key combinations; `Shift+Return` or `Shift+Enter`. However,

`Ctrl+J` or `Ctrl+M` key combinations will continue to insert a newline in this case also.

Note: The settings of the *Enter Key Executes Command* only apply if the *Syntax Highlighting* check box is selected. For more information, see "Typing SKILL Commands" in the "Using the Command Interpreter Window" chapter.

9. Select one or both of the *Warning* and *Error* check boxes in the *Raise CIW On* field. This field specifies whether the CIW should be moved to the front when a *Warning* and/or an *Error* has been displayed in the log window. The default is not to move the CIW to the front in either of these situations.

   The following environment variables can also be used:

   ```
   ui raiseCIWonError boolean t/nil

   ui raiseCIWonWarning boolean t/nil
   ```

10. Click *OK* to close the form (or *Apply* if you do not want to dismiss the form). Your settings are applied only to this session.

Save your settings and apply them to future sessions by using the following steps:

1. In the Command Interpreter Window (CIW), choose *Options – Save Defaults*.

   The Save Defaults form appears.

2. Click *OK*.

   The program saves these settings to your `.cdsenv` file and applies them to future sessions (see "Specifying New Default Values" on page 216).

Specifying Dashboard Controls

You can control the display of dashboard indicators so that you can either see them only in the *CIW* or on both the *CIW+Session* windows. You could also have them *Hidden* by selecting *Options – User Preferences* in the CIW.

This will display the User Preferences form.

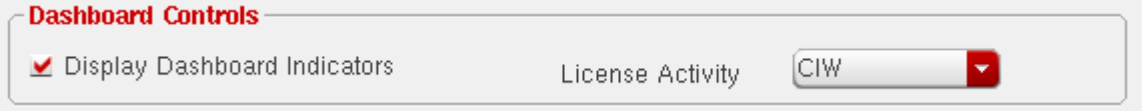


Figure 6-1 Dashboard Controls

Specifying , Library Browser, and CIW Preferences

You use thePreferences form for the following tasks:

- Specifying the Size of the Recently-Used List on page 142
- Specifying Auto Save Settings on page 143
- Specifying When the Library Browser Form Appears on page 144
- Specifying When the Exit Prompt Appears on page 145

Tip

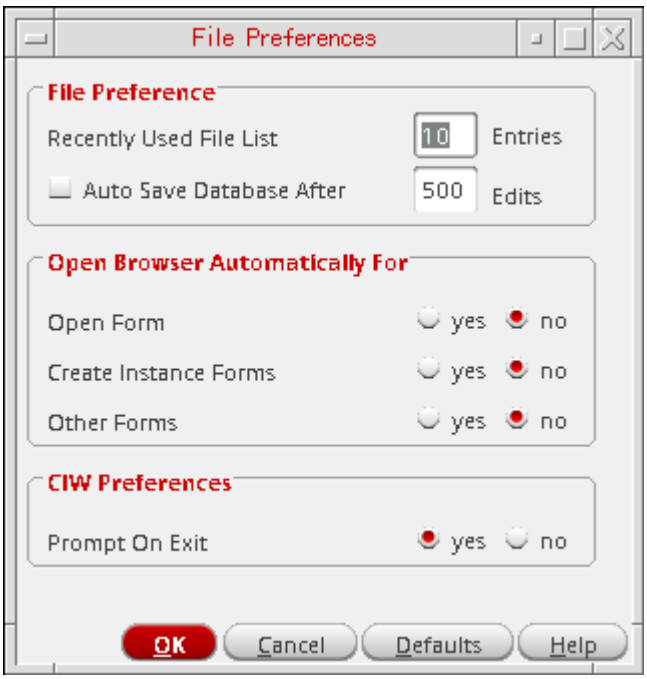
Use *Options – Save Defaults* to apply your changes to future sessions. See Specifying New Default Values for more information.

Specifying the Size of the Recently-Used List

To specify the size of the recently-used file list that appears on the Command Interpreter Window (CIW) menu, do the following:

1. In the CIW, choose *Options – Preferences*.

   The Preferences form appears.



2. In the *Recently used file list* field, type the maximum number of files you want to see on this list. The maximum is 20.
3. Click *OK*.

   The program applies your settings immediately; you do not have to restart the software.

Tip

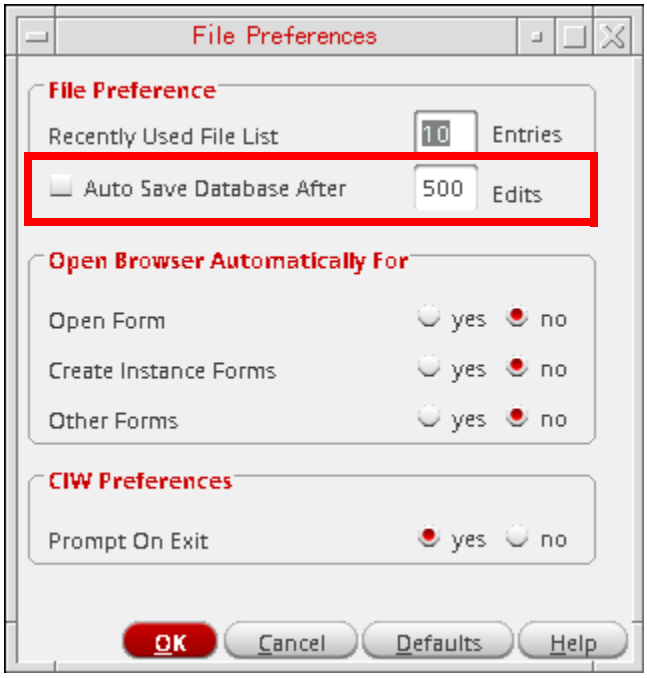
Use *Options – Save Defaults* to apply your changes to future sessions. See Specifying New Default Values for more information.

Specifying Auto Save Settings

To activate, and specify, auto save settings for database edits, do the following:

1. In the CIW, choose *Options – Preferences*.

   The Preferences form appears.



2. Check the *Auto Save Database After* check box.
3. Optionally, edit the number of database edits required before an auto save is performed. The default is 500.

   Note: The minimum acceptable edit value is 100, and the maximum is 9999.

4. Click *OK*.

   The program applies your settings immediately; you do not have to restart the software.

Use *Options – Save Defaults* to apply your changes to future sessions. See Specifying New Default Values for more information.

Specifying When the Library Browser Form Appears

To specify when the Library Browser form appears, do the following:

1. In the CIW, choose *Options – Preferences*.

   The Preferences form appears.

2. In the *Open Browser Automatically For* group box, select *yes* for those occasions when you want the Library Browser to appear automatically.

   - *Open Form* opens the Library Browser form along with the Open form (when you open additional design views)
   - *Create Instance Forms* opens the Library Browser form along with any Add or Create forms that you use to place a component instance in a design
   - *Other Forms* opens the Library Browser form along with other forms (such as the Open form) that you use to open design views or add components

3. Click *OK*.

   The program applies your settings immediately; you do not have to restart the software.

Tip

Use *Options – Save Defaults* to apply your changes to future sessions. See Specifying New Default Values for more information.

Specifying When the Exit Prompt Appears

To specify whether an Exit prompt appears when you choose – *Exit* from the CIW menus, do the following:

1. From the CIW, choose *Options* – *Preferences*.

   The Preferences form appears.

2. In the *CIW Preferences* group box, select one of the following *Prompt On Exit* options:

   - Select *no* if you do not want Exit prompts to appear.
   - Select *yes* if you do want Exit prompts to appear.

3. Click *OK*.

   The program applies your settings immediately; you do not have to restart the software.

Tip

Use *Options – Save Defaults* to apply your changes to future sessions. See Specifying New Default Values for more information.

Note: You can also specify whether the Exit prompt appears by putting one of the following lines in your `.cdsenv` file:

`ddserv.ciw promptOnExit boolean nil`	The Exit prompt does not appear.
`ddserv.ciw promptOnExit boolean t`	The Exit prompt appears.

This setting takes effect the next time you run the software.

Saving and Restoring Window Positions

See the following topics:

- Saving Window Positions (next)
- Restoring Window Positions on page 147
- Saving and Restoring the Position of the CIW Using SKILL on page 147

Saving Window Positions

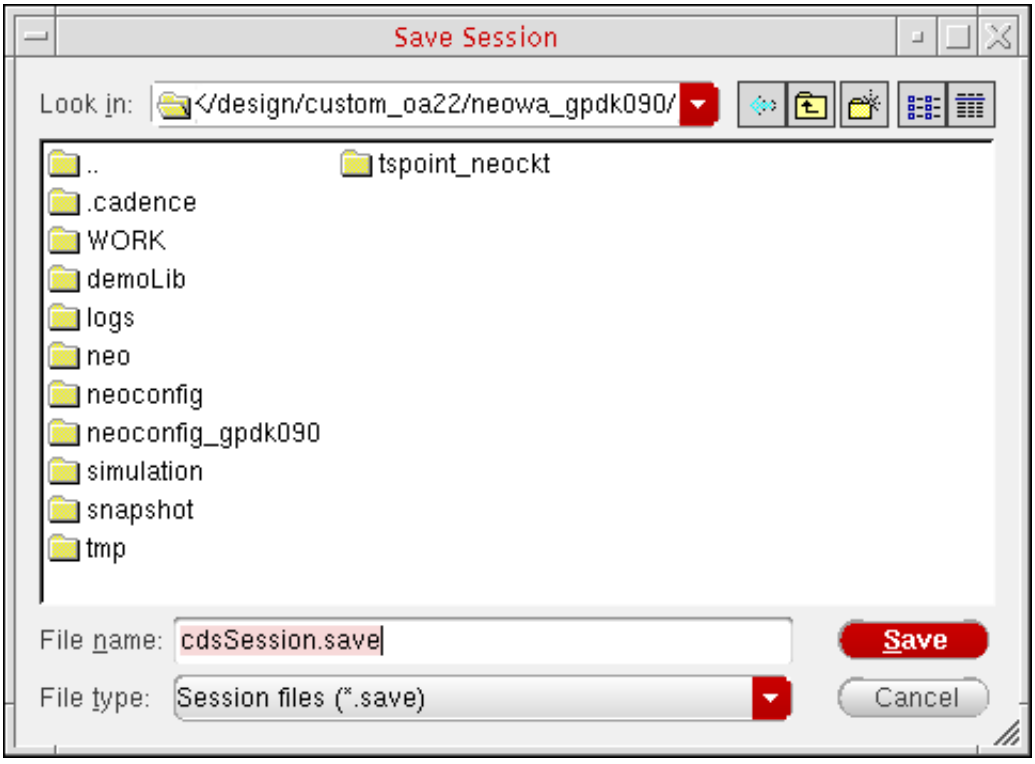
You can save the current position of design windows and forms. The exact information saved depends on your application.

Note: This does apply to workspace positioning.

To save window and form positions, do the following:

1. From the CIW, choose *Options – Save Session*.

   The Save Session form appears.



The default location appears in the Look in field. The default name appears in the name field. The *type* is *Session files (*.save)*.

2. (Optional) In the *Look in* field, change the directory location for the saved session file.
3. (Optional) In the *name* field, type a different name for the saved session file.
4. Click *OK*.

   The program saves session settings to the specified file.

Restoring Window Positions

To restore the window and form positions saved from a previous session, do the following:

➤ Start the Cadence software using the `-restore` option as follows:

```
startupCmd -restore session
```

where *`startupCmd`* is your startup command and *`session`* is the session file name that you saved previously.

For example, to start the `virtuoso` binary and restore the settings saved in the default `cdsSession.save` file, type the following:

```
virtuoso -restore cdsSession.save
```

Saving and Restoring the Position of the CIW Using SKILL

To save and restore the position of the CIW, do the following:

1. Move the CIW to the desired location and resize it the way you want it.
2. Note the `hiResizeWindow` command that the program writes to your `CDS.log` file:

   ```
   cat ~/CDS.log
   ```

   For example:

   ```
   \a hiResizeWindow(window(1) list(532:0 1257:193))
   ```
3. Type the `hiResizeWindow` command as it appears (everything after the `\a`) in your `.cdsinit` file to cause the program to move and resize the CIW according to these settings the next time you run it.

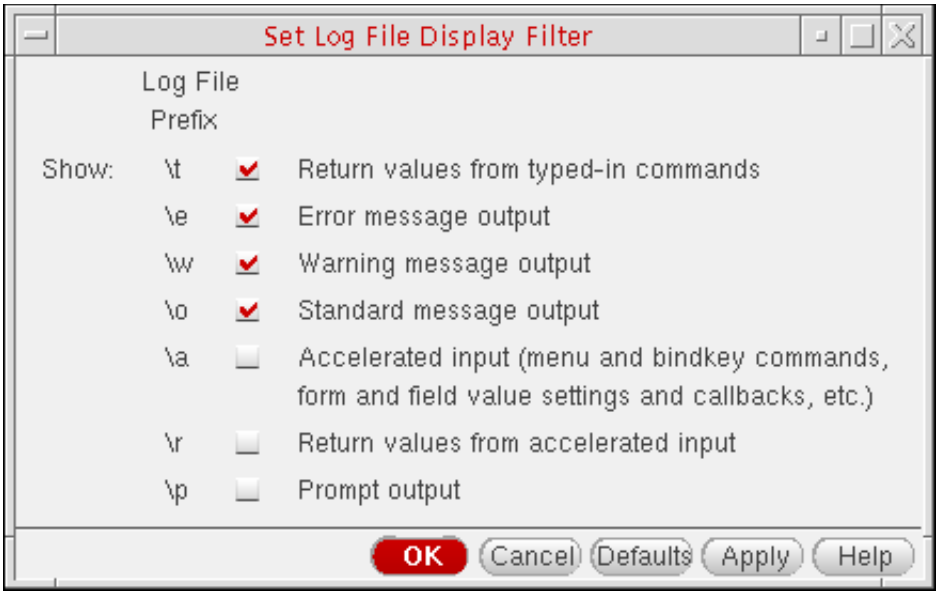
Changing the Log Filter Options

By default, only error messages, warning messages, program results, and results of running a function appear in the output area of the Command Interpreter Window (CIW). All other log file contents are filtered out.

To specify what information appears in the output area of the Command Interpreter Window (CIW), do the following:

1. Choose *Options – Log Filter*.

   The Set Log Display Filter form appears.



   A check box for each item type appears on the form. The backslash-character that appears at the beginning of the line in the `CDS.log` file for each item type appears to the left of each check box. (One backslash-character you might see in the `CDS.log` file for which there is no check box on this form is `\t` for typed input: Typed input always appears in the output area.)

2. Check or uncheck each check box accordingly to indicate what items you want to see in the output area

3. Click *OK*.

   The text in the CIW output area is updated to display the items you specified.

To apply these settings to subsequent sessions, type these changes in your `.cdsenv` file.

Note: The prefix `\#` is used in the `CDS.log` file to filter out statistic information such as output memory use and X resource ID.

Customizing the Menu Banner

You can customize the menus that appear on the menu banner and the menu commands that appear on each menu according to your needs. You can define banner menus in the CIW and other applications for which you have a license. Menu files for the Virtuoso Design Environment are installed in

your_install_dir/tools/dfII/etc/tools/menus/*binaryName*.menus

The following file explains the syntax for menu definitions:

your_install_dir/tools/dfII/etc/tools/menus/README

- For more information on customizing the menus in the CIW, see Customizing CIW Menus in the Design Data Services section of the *Virtuoso Design Environment SKILL Reference*.
- For information on customizing pull-down, pop-up, and object-sensitive menus in the Virtuoso Schematic Editor see *Customizing the Virtuoso Schematic Editor.*
- For more information on the function to customize menu banners in ADE Verifier, see *Customize Menu Banners*.
- For more information on customizing menus in ADE Explorer, see *Customizing Menus*.

Customizing Toolbars

You can choose to customize existing Virtuoso application toolbars or create new toolbars containing your own preferred functionality.

You can find application-specific toolbar definitions in text files and stored in *your_install_dir*/share/cdssetup/dfII/toolbars/byApplication.
For example, toolbars specific to the Virtuoso Layout Suite are stored in the following file:

your_install_dir/share/cdssetup/dfII/toolbars/byApplication/Layout.toolbars

Whenever you launch an application, the program locates the toolbar definitions file and creates toolbars accordingly.

You can create custom toolbar definition files and put them in your local .cadence hierarchy so that the program finds and uses these definitions first. See Understanding the .cadence Hierarchy for more information.

See also

- Toolbar Definition File Format on page 152
- Toolbar Definition Search Path and Customization on page 161

Toolbar Definition File Format

The toolbar definition file is an ASCII text file containing toolbar definition information.

Here is an excerpt from `Layout.toolbars`:

```
(
    (
    nil
    name le Toolbar
    text " "
    items (
        (
        nil
        type     action
        name     le ToolbarOpen
        text     "Open"
        icon  "file-open.png"
        callback "de Open()"
        disabled t
        )
        (
        nil
        type     action
        name     le ToolbarSave
        text     "Save"
        icon  "file-save.png"
        callback "geSave()"
        enableCondition modified
    )
)
...
```

Here is a snippet to include `subAction` in the `actionList`.

```
(
        nil
        type    action
        subType subAction
        name    lebROGSorV
        text    "Shapes Or Vias"
        icon "shapes-vias.png"

        )
        (
        nil
        type    action
        subType subAction
        name    lebROGWire
        text    "Entire Wire"
        icon "wire-entire.png"
```

```
)
(
nil
type    action
subType subAction
name    lebROGConnect
text    "Connected Shapes"
icon "shapes-connect.png"
)
(
nil
type    action
subType subAction
name    lebROGShapesOnNet
text    "All Shapes on the Net"
icon "shapes-on-net.png"
)
(
nil
type    action
subType subAction
name    lebROGNet
text    "Net"
icon "net-category.png"
)
(
nil
type    action
subType actionList
name    lebROG
text    "Selection Granularity"
icon "shapes-vias.png"
actionList(lebROGSorV lebROGWire lebROGConnect lebROGShapesOnNet
lebROGNet)
enabled nil
)
```

The toolbar name (such as `le Toolbar`) must be unique across all applications. The text the program uses for the toolbar title and tooltip text (such as `" "`) need not be unique.

Here is a further example from `Symbol_XL.toolbars`:

```
(
```

```
    (
        nil
        inheritToolbarsFrom "Symbol"
    )
    (
        nil
        name symSearchToolbar
        text "Search"
        items ( )
    )
)
```

Note: The `inheritToolbarsFrom` option allows you to include the toolbars from another application at any point in a toolbar file (between toolbar definitions).

See also:

- Format for Main Toolbar Definition on page 154
- Format for Item Type action on page 155
- Format for Item Type comboBox on page 157
- Format for Item Type typein on page 159
- Format for Item Type separator on page 159

Format for Main Toolbar Definition

Mandatory Fields

`name`	The name to be used in the creation of the toolbar. The name must be unique. **Note:** The Cadence Design Environment will automatically add *`_sessionWindowNumber`* to the name.
`text`	The string to be used in the toolbar title and *ToolTip*.
`items`	The list of item descriptions for the toolbar.

Optional Fields

`invisible`	When set to `t` this will ensure that the toolbar is initially hidden on application launch.

`toolButtonStyle`	Specifies one of the following toolbar button styles: `textOnly`, `iconOnly`, `textBesideIcon`, or `textUnderIcon`. **Note:** The default is `iconOnly`.
`getActionFunction`	Specifies a function that is called on all toolbar items of type `'getAction` on this toolbar. The specified function must return an appropriate `hiAction` value. This functionality lets you reuse the existing `hiAction` values and share them amongst widgets. Therefore, if an `hiAction` is toggled using a widget, all other widgets that use the same underlying `hiAction` will automatically reflect the correct state.

Format for Item Type action

Mandatory Fields

`type`	A direct `action` (callback).
`name`	The name to be used in the creation of this `action`. **Note:** The name must be unique among all `actions`.
`text`	The text used for the `iconText` and tooltip text.
`callback`	The SKILL string to be actioned when the `action` is processed.
`icon`	The name of the icon file (must be in `.png` format). This file is looked up in the standard `setup.loc` locations, prefixed by the path `icons/24x24`. The standard locations include `<installdir>/share/cdssetup`, `./cadence` and `~/.cadence`. See also "Toolbar Definition Search Path and Customization" on page 161.

`enableCondition`

Sets the toolbar button enablement condition, where the following activation settings can be specified:

- `selected`: object is selected, button is enabled.
- `modified`: cellview has been modified, button is enabled.
- `editable`: cellview is editable, button is enabled.
- `assistantsPresent`: assistants exist in cellview, button is enabled.
- `notMultiSheet`: cellview is not part of a multi sheet, button is enabled.
- `modifiedNotScratch`: identifies that the cellview is modified and not a scratch cellview.

It can be a:

- Symbol of a function that is called with the window and item as arguments.
- Symbol that cannot be called, but is evaluated.
- List whose first member is a function.

The enableCondition can be set only for action types.

`visibleCondition`

Sets the visibility condition for a toolbar. Settings that can be specified are comboBox, typein, and separator.

It can be a:

- Symbol of a function that is called with the window and item as arguments.
- Symbol that cannot be called, but is evaluated.
- List whose first member is a function.

The visibleCondition can be set for any toolbar item type.

Optional Fields

`invisible`

When this field is included and set to `t`, the `action` item will initially be hidden.

`subAction`	What this `action` will be used for. Choose from: ■ `subAction`: the `action` will be used in a subsequent `actionList` (a menu of actions), or ■ `actionList`: the `action` has an action list; a menu of actions.
`disabled`	When this field is included and set to `t`, the action will initially be disabled.
`checkable`	When set to `t` this allows the use of the `checked` attribute.
`checked`	When `checkable` is set to `t`, also setting this attribute to `t` will make the `action` initially checked.
`editable`	The action is enabled when the cellview is editable.
`modified`	The action is enabled when the cellview has been modified but not saved.
`selected`	The action is enabled when the cellview is editable and at least one object is selected.
`assistantsPresent`	The action is enabled when the workspace contains at least one assistant. The action is enabled even if all the assistants are hidden.
`notMultiSheet`	The action is enabled when the property `schType` is not set to "index". A schematic with the property `schType` set to "index" means that it is the index of a multi-sheet schematic.
`tootltip`	You can specify tooltips for user-defined toolbars. These tooltips are displayed in the status bar of the application.

Format for Item Type comboBox

Mandatory Fields

`type`	`comboBox`
`name`	The name of the `comboBox`.
`tooltip`	The text used for the `comboBox` *Tooltip*.

`items`	A list of strings, a string, or a SKILL expression that form the items in the `comboBox`. ■ If items is a list of strings, the items of the comboBox is assigned the list as is. ■ If items is a string and the string is a callable function name, the function is applied without arguments, and the items of the `comboBox` is assigned the value of the function. ■ If items is a SKILL expression, it is evaluated and the items of the `comboBox` are assigned the results of the evaluation. If evaluating the SKILL expression results an error, items is assigned `nil`.
`callback`	The SKILL string to be called.
`width`	The width to be given to `hiCreateToolbarComboBox`.
`value`	The value of the `comboBox`, which could be a string or a SKILL expression. If value field is a string, the value of the `comboBox` is assigned the string as is. If it is an SKILL expression, the expression is evaluated and the value of the `comboBox` is assigned the result of the evaluation. If evaluating the expression results in an error, value is assigned "`Error`".

Optional Fields

`invisible`	When this field is included and set to `t`, the `comboBox` item will initially be hidden.
`disabled`	When this field is included and set to `t`, the `comboBox` will initially be disabled.
`editable`	The action is enabled when the cellview is editable. Default is `t`.
`prompt`	A string that can be used to prompt user action. Default is `""`.

Format for Item Type typein

Mandatory Fields

`type`	`typein`
`name`	The name of the `typein`.
`tooltip`	The text used for the `typein` *Tooltip*.
`callback`	The SKILL string to be called.
`value`	The value (string) of the `typein`.
`width`	The width to be given to the `typein`.

Optional Fields

`invisible`	When this field is included and set to `t`, the `typein` item will initially be hidden.
`disabled`	When this field is included and set to `t` the `typein` will initially be disabled.
`editable`	The action is enabled when the cellview is editable. Default is `t`.
`prompt`	A string that can be used to prompt user action. Default is `""`.

Format for Item Type separator

Mandatory Fields

`type`	`separator`
`name`	The name of the `separator`.

Optional Fields

`invisible`	When this field is included and set to `t`, the `separator` item will initially be hidden.

Format for item type inheritToolbarsFrom

Mandatory Fields

`inheritToolbarsFrom`	The string name of an application whose toolbars will be inserted in this position in this application's toolbar.

Toolbar Definition Search Path and Customization

The program searches for toolbar definition files in locations specified in your `setup.loc` file. The default location for application-specific toolbar definition files is:

your_install_dir`/share/cdssetup/dfII/toolbars/byApplication/`*appName*`.toolbars`

You can install your customized toolbar definition files so that the software finds those first.

For more information about the `setup.loc` file, see the Cadence Setup Search : setup.loc in *Cadence Application Infrastructure User Guide*.

Using Toolbar Manager

Toolbar Manager lets you make incremental and local changes to the Virtuoso application toolbars. You can add, edit, or remove toolbars and toolbar items of applications whose toolbars are defined in Virtuoso Design Editor toolbar files.

Note: Toolbar Manager cannot modify toolbars created programmatically using SKILL APIs, such as `hiCreateToolbar`.

Toolbar Manager saves your changes in the *`applicationName`*`.overlay` files in the `.cadence` directory present in your home directory. When you launch an application, the Virtuoso design environment creates toolbars based on the *`applicationName`*`.toolbars` file, overlaid by the *`applicationName`*`.overlay` file. This method ensures the seamless implementation of your changes even after you upgrade the Virtuoso installation.

For details on the `.cadence` hierarchy, see Understanding the .cadence Hierarchy.

The `.overlay` files follow the same filename convention as the `.toolbars` files. For example, the `.toolbars` file of Schematic Editor is `Schematics.toolbars`. The overlay file of this application is `Schematics.overlay`.

Note: Virtuoso releases previous to Virtuoso IC6.1.6 do not support the overlay files.

Note: For certain conditions of invalid items in an overlay file, the respective application does not launch and an error message appears in Virtuoso CIW. You must fix the syntax using Toolbar Manager for that application.

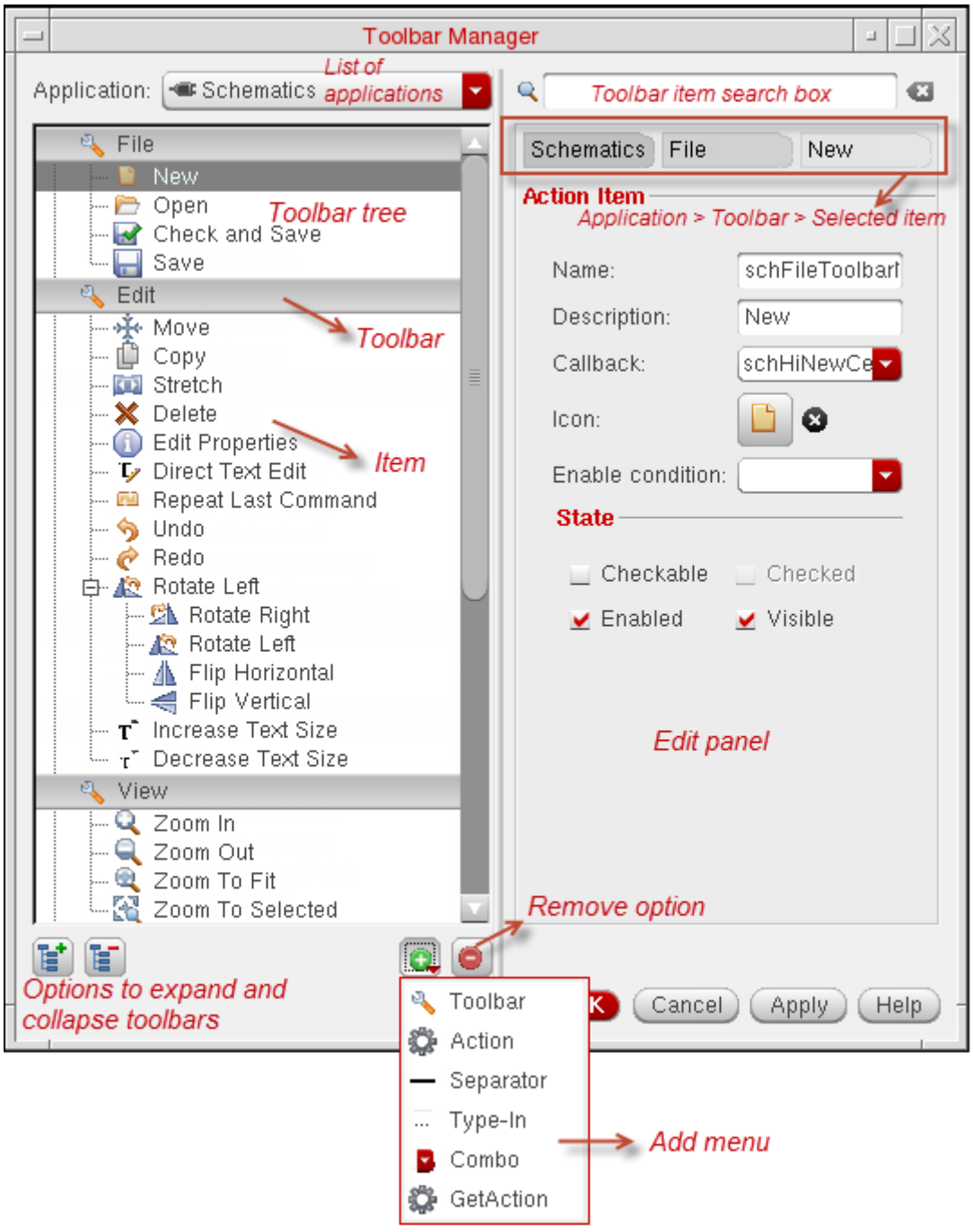
To customize toolbars of a supported application using Toolbar Manager:

1. Do one of the following:
   - Choose *Options – Toolbars* from the Virtuoso CIW.

- Choose *Window – Toolbars – Customize* from within a Virtuoso session window.
- Right-click the toolbar in the application and select *Customize*.

Note: If the *Customize* option does not appear when you right-click a toolbar, you cannot edit that toolbar using Toolbar Manager.

The Toolbar Manager form appears.



2. Select the application from the *Application* list. The toolbars of that application appear in the Toolbar tree.

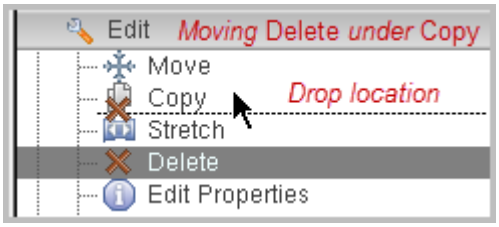
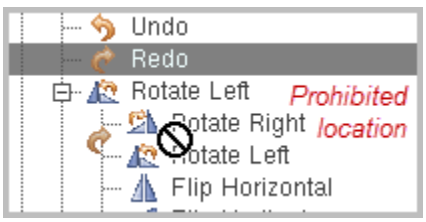
   - Toolbar Manager displays the list of applications in their hierarchical order. Sub-applications typically inherit various toolbars from the parent applications. For

example, when you add a new toolbar in Schematics, it also becomes available in Schematics XL.

- To search for any data in the toolbars of the selected application, use the *Search* box. Toolbar Manager filters the list of items based on your search string.
- You can expand or collapse the toolbars using the icons on the bottom-left corner.
- The Toolbar tree displays items with unsaved changes in blue text.

3. Change the application toolbars, as required.

To do this...	Perform these steps...
Add a new toolbar	**1.** Right-click the Toolbar tree and click *Add* or click *Add* at the bottom of the form to access the *Add* menu. **2.** Click *Toolbar*. **3.** Specify the toolbar details. For more information, see Format for Main Toolbar Definition.

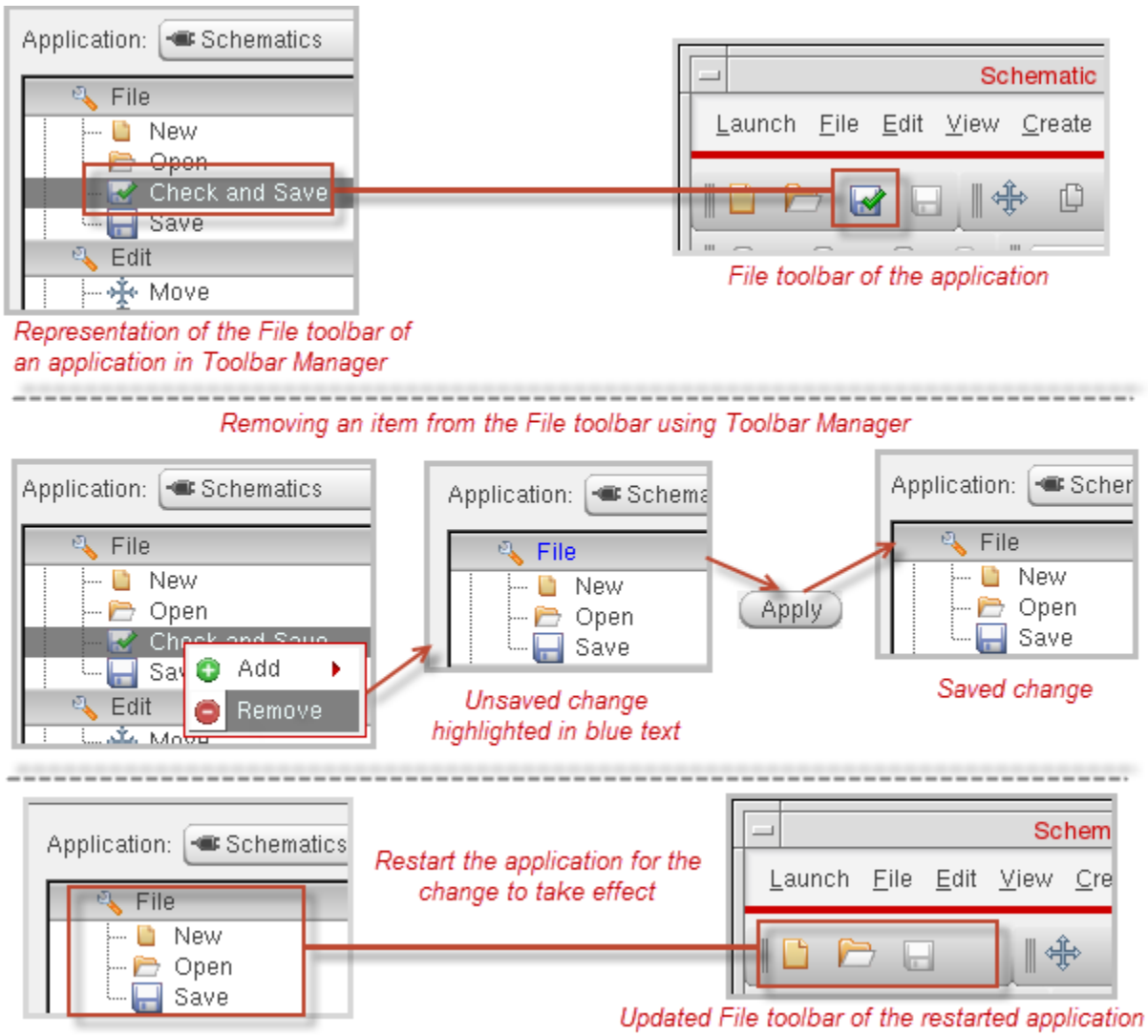
To do this...	Perform these steps...
Add a new item	**1.** Select the toolbar in the Toolbar tree. **2.** Click the *Add* button to access the *Add* menu. **3.** Select the item type. You can choose from: ❑ *Action*: Adds an action item to perform an action. ❑ *Type-in*: Adds an item with an input box. ❑ *Combo*: Adds an item with a list for selection. ❑ *Separator*: Adds a separator for differentiating between items. ❑ *GetAction*: Passes the name of an item to the action retrieval function specified on the item's toolbar. This option allows actions to be reused. **Note:** By default, the callbacks for type-in (or text) and drop-down list box fields are only triggered if the value has changed. If you want the callback to be triggered when the `Enter` or `Return` key is pressed regardless of whether the value in the field has changed, select the *Always execute callback when "Enter" key is pressed* check box. **4.** Specify the item details on the Edit panel. For details, see "Toolbar Definition File Format" on page 152.
Move an item	➡ Select the item on the Toolbar tree, drag it to the new location, and drop the item. The location where you can drop the item is illustrated by a horizontal line between the items. You cannot drop the item to a location where the line is not visible or the cursor changes to the forbidden cursor. Releasing the mouse button in a prohibited location cancels the drag operation and returns the selection to the original location.  

To do this...	Perform these steps...
Edit a toolbar or item	**1.** Select the toolbar or item on the Toolbar tree. **2.** Edit the details on the Edit panel.
Delete a toolbar or item	**1.** Select the toolbar or item on the Toolbar tree. **2.** Click the *Remove* button on the bottom-right of the Toolbar tree.

4. To save the changes without closing Toolbar Manager, click *Apply*. To save the changes and exit Toolbar Manager, click *OK*.
5. Restart the application to implement your changes.

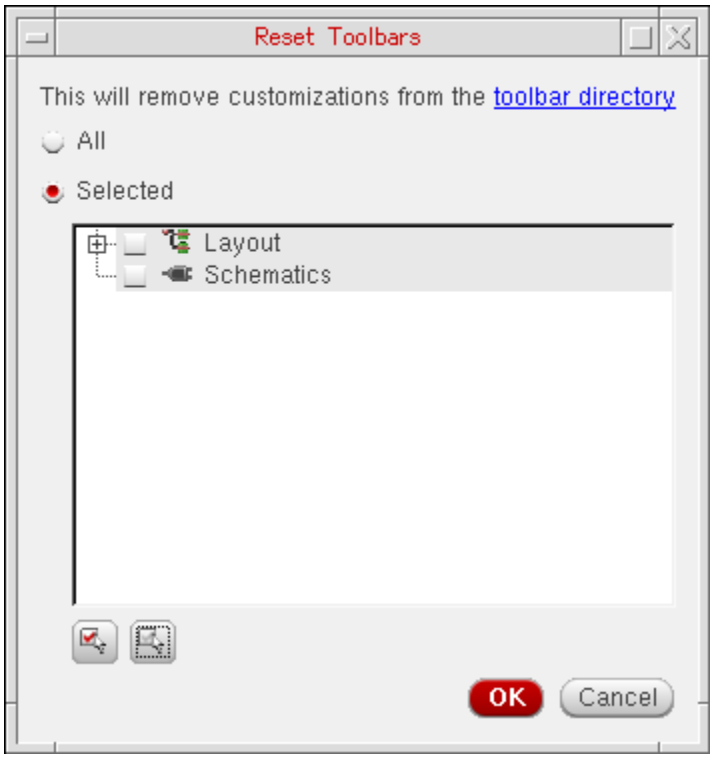
Note: Your changes will not take effect in the currently opened applications. They take effect in applications you open after saving your changes.

The following figure illustrates how you remove the *Check and Save* item from the toolbar of Schematic Editor.



Resetting Toolbar

You can revert changes by resetting the toolbars on a per-application basis. To do this, you need to click the *Reset* button on the Toolbar Manager form. Once you click the *Reset* button, the Reset Toolbars dialog box is displayed.



This dialog lists all the applications in which a toolbar or an item has been added, removed, or edited. Select the application that you need to reset by clicking the check box and click the *OK* button.

Note: To select all applications, click the *Select all* button. In addition, to deselect all applications, click the *Select none* button.

Caution

This action cannot be undone, so be sure that the correct applications are selected.

Troubleshooting Information for Toolbar Manager

Changes to the toolbar configuration are static, that is, toolbar configuration is determined at the time the window is instantiated. The behavior specified for a toolbar may be overridden dynamically through SKILL, and this behavior, if inconsistent with the static behavior, can result in confusion.

This inconsistency is typically encountered when a toolbar item that you did not define is customized—toolbar items defined by others can have side effects that might not be immediately obvious when looking at the toolbar item details in Toolbar Manager. SKILL callbacks associated with toolbar items can manipulate the toolbar item directly and change the expected behavior.

One example is the use of multiple checkable toolbar buttons which appear to function like radio buttons, where only one button can be selected at a time. In particular, when a checked item in a radiobutton group is pressed again, it may be desirable to remain checked, rather than toggling to an unchecked state. It is also typical to ensure that only one item is checked at a time, such that checking one item will uncheck any others in that group. This is typically implemented via a SKILL callback for each item—based upon which button was pressed, the callback will adjust all buttons to reflect the desired state. In this case, if one is not aware of these effects of the SKILL callback, the behavior will seem inconsistent with the behavior specified for those toolbar items.

Understanding the .cadence Hierarchy

The `.cadence` hierarchy contains local customizations for files that Cadence provide. For each location specified in the `setup.loc` file, the program first searches for `.cadence` directories.

The following structure is used for `.cadence`:

```
<directory>/.cadence/[<user>]/<product>/<tool>/[<version>]/<files>
```

Where:

- Prior to saving for the first time, the `user` directory is optional and not likely to exist in system areas. Whenever data is saved however, it will always be saved under a user-specific directory.
- The `version` number is also optional, but should always be used when saving. This setting is optional in order to support the loading of default settings from an installation hierarchy.
- For any given `.cadence` directory, the options in the user-specific subdirectory have a higher priority than the non-user-specific options.

This hierarchy structure provides for solid file management when multiple versions of Cadence tools are installed in a common directory.

Note: When CDMS (Cadence Directory Management System) is used by an application, it is expected that user data will move, for example, from 613 to 614 on first use of 614. CDMS implicitly uses `.cadence` as the top-level directory name.

Your local `.cadence` directory can contain the following file and subdirectories of information:

File or Directory	Description
`history/`*`userName`*`.history`	The program creates this file to maintain information that appears in the History tree on the ADE Explorer Data assistant pane *Important* You must not edit this file.
`Navigator`	Directory containing an `Options.xml` file where the program stores any options you specify on the Navigator Options form. *Important* You must not edit this file.
`toolbars`	Directory you can create to contain custom toolbar definition files
`workspaces`	When you save a custom workspace, the program creates this directory along with an application-specific subdirectory containing your custom `.workspace` file; your `workspaces` directory might contain several application-specific subdirectories and each application-specific subdirectory might contain several `.workspace` files **Note:** `.workspace` files are binary files that you cannot edit.
`workspace.default`	If you specified a default workspace for an application, the program writes the name of the default workspace to this file

If you save any job policies, the program creates a `jobpolicy` subdirectory in your `.cadence` directory to maintain job policy definitions.

Files and directories in the *your_install_dir*/share/cdssetup/dfII hierarchy include:

or Directory	Content
VLS-EXL	Modgen.patterns pattern file for Modgen; see Working with Patterns in the *Virtuoso Module Generator User Guide* for more information
cds.lib file	Default library definitions; see Cadence Library Definition in the *Cadence Application Infrastructure User Guide* for more information
ci	config.xml file defining constraint types for layout as well as a mapping between constraint type and parameter names and what the program displays for these; see the *Virtuoso Unified Custom Constraints User Guide* for more information
default.drf file	Default display resource settings; see the Display Resource Development and Usage chapter in the *Technology and Display Resource User Guide* for more information
toolbars	byApplication subdirectory of Cadence-provided .toolbars files
workspaces	Application subdirectories containing Cadence-provided .workspace definition files and workspace.default file; for more information, see Getting Started with Workspaces.

7

Specifying Environment Settings

You can specify environment settings and initialization instructions for Cadence applications in `.cdsenv` and `.cdsinit` files, respectively. Using these settings, you can customize your Cadence software environment. Your system administrator can copy a customized version of these files to your local site directory: *`your_install_dir`*`/tools/dfII/local`. You can also maintain local copies in your home directory. Settings in your local files take precedence over all other settings.

Cadence software loads settings in the following order:

1. `.Xdefaults`
2. `.cdsenv` (environment settings)

   Note: If you specify `.cdsenv` in your `csfLookupConfig` file, Cadence applications use the Cadence Setup Search File mechanism to find this file.

   - *`your_install_dir`*`/tools/dfII/etc/tools/`*`binaryName`*`/.cdsenv`

     This file is present for every application that supports the `.cdsenv` environment-setting mechanism.

   - *`your_install_dir`*`/tools/dfII/local/.cdsenv`

     This file is present only if your site administrator copies a customized environment file to this location.

   - search order as specified by the `CDS_LOAD_ENV` environment variable, if set; otherwise, `~/.cdsenv` (see "Copying and Editing the Default .cdsenv File" on page 187 for more information).

     Note: If the value of `CDS_LOAD_ENV` is set to CSF, then the first directory found in `setup.loc` (see also Search Mechanism in the *Cadence Application Infrastructure User Guide*) will be the default directory to which `.cdsenv` will be saved.

3. `.cdsinit` (initialization instructions)

   Note: If you specify `.cdinit` in your `csfLookupConfig` file, Cadence applications use the Cadence Setup Search File mechanism to find this file.

- *your_install_dir*/tools/dfII/local/.cdsinit

  Note: This file is present only if your site administrator copies a customized environment file to this location.

- .cdsinit in your current working (project) directory
- ~/.cdsinit (see Copying and Editing the Default .cdsinit File)

Cadence provides samples of these files in your installation hierarchy:

your_install_dir/tools/dfII/samples/.cdsenv
your_install_dir/tools/dfII/cdsuser/.cdsinit

Note: You can also customize parts of your environment using the *Options* menu in the Command Interpreter Window (CIW). See Chapter 6, "Customizing Your Design Environment" for more information.

Each file has a different format:

- The .cdsenv file contains settings of the form

  tool[.*section*] *setting type value*

  For more information about the .cdsenv file, see the following topics:

  - Copying and Editing the Default .cdsenv File on page 187
  - Specifying a Search Order for .cdsenv on page 195
  - Specifying User Interface Preferences in .cdsenv on page 197
  - Specifying Default Application in .cdsenv on page 210
  - Specifying License Checkout Behavior in .cdsenv on page 212

- The .cdsinit file contains executable commands for custom software initialization (such as bindkey definitions, custom SKILL procedures, and user preferences)

  For more information about the .cdsinit file, see the following topics:

  - Copying and Editing the Default .cdsinit File on page 175
  - Setting the SKILL Search Path on page 176
  - Specifying the Editor on page 176
  - Specifying User Preference Options on page 177
  - Specifying Log Filter Options on page 179

Copying and Editing the Default .cdsinit File

To change your Cadence startup environment, you can create a `.cdsinit` file in your home or working directory by copying the default file from the `cdsuser` subdirectory of the Virtuoso Design Environment installation directory as follows:

1. On the input line of the Command Interpreter Window (CIW), type the following command and press *Return*:

   ```
   println(getInstallPath())
   ```

   The directory where the Cadence software is installed appears in the output area.

2. In a terminal window, type the following command and press *Return*:

   ```
   cd
   ```

   Your current working directory is your home directory.

3. Type the following command and press *Return*:

   ```
   cp dfII_InstallDir/cdsuser/.cdsinit .
   ```

   where *dfII_InstallDir* is the directory path returned by the command you typed on the input line of the CIW in step 1 above.

Once you have a valid `.cdsinit` file in your home directory, you can edit the file as follows:

1. Open the `.cdsinit` file with a text editor.

   For example, to use `vi`, type the following command:

   ```
   vi .cdsinit
   ```

2. Make the changes you want.

3. Save the file and exit the editor.

   The program uses these settings the next time you run the software.

To apply these settings to the currently running session, you can load the modified `.cdsinit` file as follows:

➤ On the input line of the CIW, type the following command and press *Return*:

```
load("~/.cdsinit")
```

The command and its result (`t` for successful execution) appear in the output area.

Setting the SKILL Search Path

A search path is a list of directories the software searches for various files, libraries, and commands. The SKILL search path, which identifies the location of your SKILL files and commands, must include directories with read-write access. Cadence software looks for SKILL files by searching the directories in the order listed in the SKILL search path.

The default SKILL search path checks for files in the following locations in the order shown:

1. *your_install_dir*/tools/dfII/local
2. The current directory, represented by a dot (.)
3. Your home directory, represented by a tilde (~)

To change your SKILL search path, do the followng:

➤ Type a list of space-separated paths as arguments to the `setSkillPath` command in your `.cdsinit` file. For example:

```
setSkillPath(". your_install_dir/tools/dfII/local ~/myskill")
```

This command instructs Cadence software to check the current directory first, then *your_install_dir*/tools/dfII/local, then the `myskill` subdirectory in your home directory.

To add another directory to the end of the current SKILL search path, do the following:

➤ Type the `setSkillPath` command in your `.cdsinit` file using the following format:

```
setSkillPath($PATH newPath sometimesPath)
```

This command sets the SKILL path to search the existing path first (the contents of the `$PATH` variable) and then the additional paths (*newPath*, then *sometimesPath*).

Specifying the Editor

You can specify the editor you want to use in Cadence text entry windows. The default editor is `vi`. For example, to use the `emacs` editor instead of the `vi` editor, do the following:

➤ Add the following line to your `.cdsinit` file:

```
editor = "xterm -e emacs"
```

Specifying User Preference Options

You can set various options in your `.cdsinit` file using the commands in the following table. Some of these options are also on the User Preferences form (see "Specifying User Preferences" on page 127).

Important

The `hiGetCIWindow` command (which returns the window identity of the CIW) must appear in your `.cdsinit` before any of the `ciw->` commands in the table that follows:

```
ciw = hiGetCIWindow()
```

To change …	Type this in .cdsinit or on the CIW input line:
Scroll bars	`ciw->useScrollbars = {t|nil}`
Infix	`ciw->infix = {t|nil}`
Option forms displayed when commands start	`ciw->expertMode = {t|nil}`
Undo limit	`hiSetUndoLimit(x_limit)` Valid Values: Integer between `0` and `128`, inclusive, where `0` disables undo
Nest limit	`ciw->nestLimit = x_nestLimit` Valid Values: Integer between `1` and `20`, inclusive Default: 5
Double-click time	`hiSetMultiClickTime(x_msec)` Valid Values: Integer number of milliseconds
Display mouse bindings	`ciw->displayMouseBinding = {t|nil}`

Text font	`hiSetFont("text" "t_fontName")`
	Valid Values: Valid font name
	Note: You can use `*` as a wildcard in any position. For example:
	`hiSetFont("text" "-*-courier-medium-r-*-*-12-*")`

Focus to cursor	`ciw->focusToCursor = {t|nil}`

To change …	Type this in .cdsinit or on the CIW input line:
Form placement	`ciw->formPlacement = {"top"|"bottom"|"left"|"right"|"center"|"default"}`
Form placement relative to	`ciw->formRelativeTo = {"CIW"|"screen"|"currentWindow"}`
Option form placement	`ciw->optionFormPlacement = {"top"|"bottom"|"left"|"right"|"center"|"default"}`
Option form placement relative to	`ciw->optionFormRelativeTo = {"CIW"|"screen"|"currentWindow"}`
Warp pointer to dbox	`ciw->warpPointer = {t|nil}`
Show menu bindkeys	`ciw->displayMenuBindkeys = {t|nil}`

Specifying Log Filter Options

You can set log filter options by placing the following command in your `.cdsinit` file:

```
hiSetFilterOptions( b_inputMenuCommands b_inputPrompts b_outputProgramResults
b_outputMenuCommands b_outputUser b_messageErrors b_messageWarnings)
```

The arguments are defined below.

b_inputMenuCommands
: Shows accelerated input in the CIW.

  Valid Values: `t` (on), `nil` (off)

b_inputPrompts
: Shows prompts in the CIW.

  Valid Values: `t` (on), `nil` (off)

b_outputProgramResults
: Shows program output in the CIW.

  Valid Values: `t` (on), `nil` (off)

b_outputMenuCommands
: Shows accelerated output in the CIW.

  Valid Values: `t` (on), `nil` (off)

b_outputUser
: Shows typed output in the CIW.

  Valid Values: `t` (on), `nil` (off)

b_messageErrors
: Shows error messages in the CIW.

  Valid Values: `t` (on), `nil` (off)

b_messageWarnings
: Shows warning messages in the CIW.

  Valid Values: `t` (on), `nil` (off)

For example, if you want to see all the information in the output area of the CIW (as seen on the Set Log File Display Filter form), add the following line to your `.cdsinit` file:

```
hiSetFilterOptions( t t t t t t t)
```

To see only output and errors, change the input menu and prompts arguments and the warning message argument to nil:

```
hiSetFilterOptions( nil nil t t t t nil)
```

For more information, see “Transcript Session” and `hiSetFilterOptions` in the *Cadence User Interface SKILL Reference*.

Viewing the Font List

To view the list of fonts available on your system, do the following:

➤ Type the following command in an X window:

```
fc-list | sort | more
```

You can also view the Set Fonts form to display and edit the current fontType settings by either selecting *Options – Fonts* from the CIW menu bar, or entering "`hiSetFont()`" in the CIW entry field:

Figure 7-1 The Set Fonts Form

For more information, including descriptions of the above font type options, see `hiSetFont` in the *Cadence User Interface SKILL Reference*.

Setting Fonts using the .cdsinit File

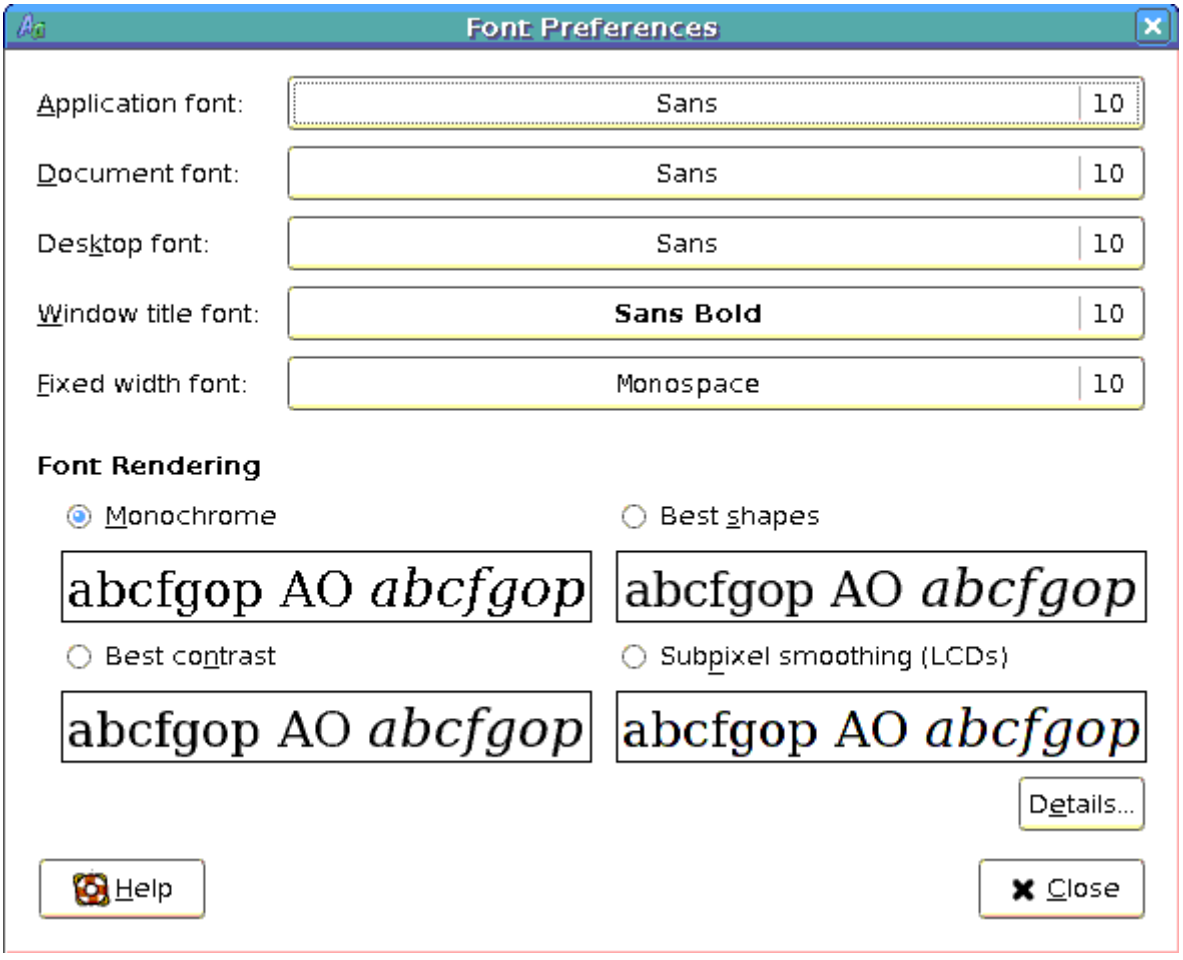
Starting with ICADV12.2, XFonts are no longer supported in the user interface (hi*) functions. XFonts have been replaced by TrueType fonts that improve the general appearance and readability of the user interface.

Note: Bitmap XFonts are still supported for Directed Acyclic Graph (DAG*)* interface and Display List (DLIST).

While TrueType fonts support anti-aliasing, font rendering is controlled using the desktop controls.

GNOME Desktop Environment

If you are using the GNOME desktop environment, you can render the fonts by choosing *System – Preferences – Fonts*. The Font Preferences dialog box is displayed.



In this dialog box, you can select the *Monochrome* option to render the fonts using the non-anti-aliased method. To render the fonts using the anti-aliased method, select any of the following options:

- *Best shapes*
- *Best contrast*
- *Subpixel smoothing (LCDs):* It is recommended to use this option for best results.

Note: In case you are using TigerVNC, you can use the preferred encoding options to render fonts. The available preferred encoding options are:

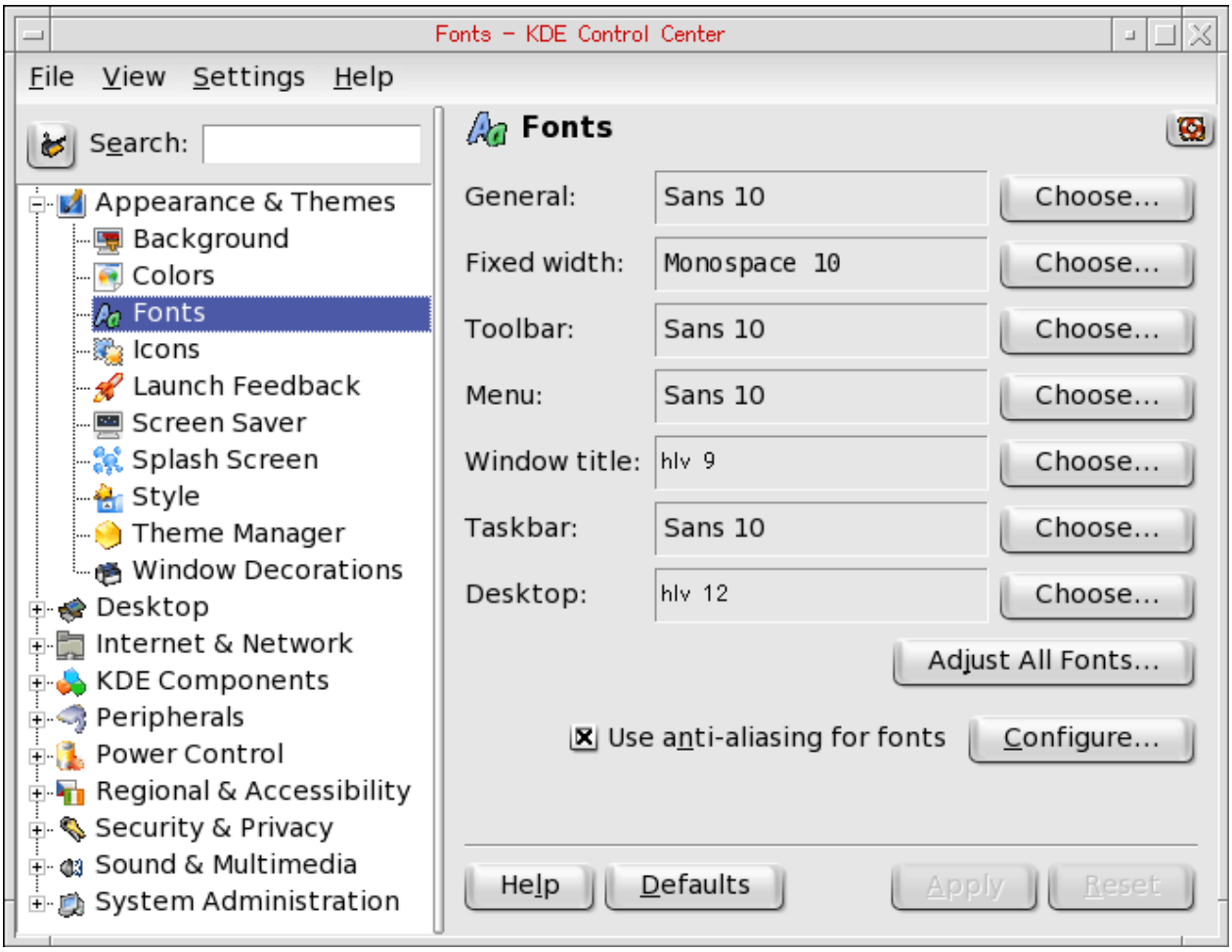
- `Tight` (Default)
- `ZRLE`
- `Hextile`
- `Raw`

For best results, you can choose `ZRLE`.

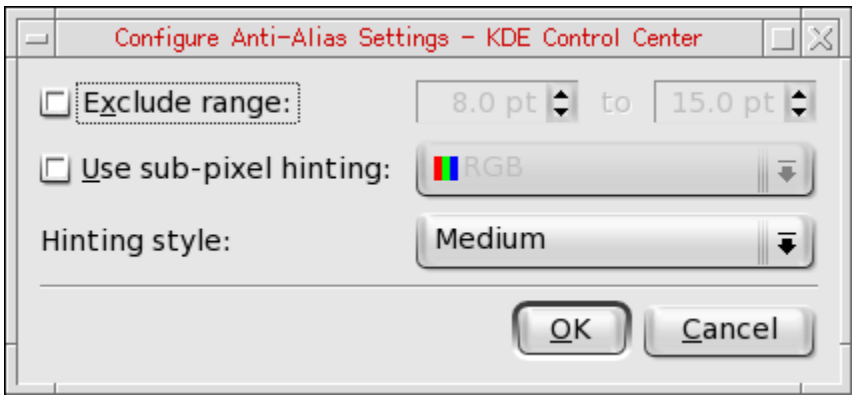
In RealVNC, the `ZRLE` preferred encoding option is selected by default.

KDE Desktop Environment

If you are using the KDE desktop environment, you can render the fonts by choosing *Appearance & Themes – Fonts* from the KDE Control Center window.



Then, click the *Configure* button to open the Configure Anti-Alias Settings – KDE Control Center dialog box.



To render the fonts using the anti-aliased method, select the *Use sub-pixel hinting* check box and then, select one of the following options from the *Hinting style* drop-down list:

- *None*
- *Slight*
- *Medium*
- *Full:* It is recommended to use this option for best results.

Caution

Ensure that the Exclude range check box is not selected. This option blocks anti-aliasing for almost all the commonly used TrueType font sizes.

Important

If you customize Virtuoso fonts with settings in the `.cdsinit` file in earlier releases, and want to have similar customizations in ICADVM20.1, you must update your `.cdsinit` file to make use of the SKILL constant, `'hicAntiAliasedFonts` to identify the TrueType font support, as shown below:

```
if( boundp( 'hicAntiAliasedFonts ) then
    hiSetFont( "ciw" ?size 14)
else
    hiSetFont( "ciw" "-*-courier-medium-r-*-*-14-" )
)
```

The above example sets the font size in CIW to 16 pixels (Default: 11). However, if you are using a version earlier than ICADV12.2, the software will run the `else` portion of the

example and set the font size to 14 pixels. Similarly, you can also change the `text` and `label` fonts using the same SKILL constant.

The RENDER extension on the X server is required for proper font rendering and drawing operations. If the current display does not have the RENDER extension, this will result in poor font appearance and sub-optimal drawing performance. The following message to this effect will be displayed when the tool is started:

```
*Warning* The RENDER extension is missing on display <DISPLAY>.
Enable the RENDER extension on the X server to improve performance and font
appearance.
```

You can use the following command to check if a particular display has the RENDER extension:

```
xdpyinfo [-d <DISPLAY>] | grep RENDER
```

Note: If checking a display other than the current display (`$DISPLAY`), specify this via the '`-d <DISPLAY>`' argument to `xdpyinfo`.

If RENDER is present, this should print `RENDER`; if it is absent, nothing will be printed.

To resolve this situation, enable the RENDER extension in the X server configuration and restart the X server.

Copying and Editing the Default .cdsenv File

To customize environment settings for Cadence software, you can create a `.cdsenv` file in your home or working directory by copying the default file from the `samples` subdirectory of the Virtuoso Design Environment installation directory as follows:

1. On the input line of the Command Interpreter Window (CIW), type the following command and press *Return*:

   ```
   println(getInstallPath())
   ```

   The directory where the software is installed appears in the output area.

2. In a terminal window, type the following command and press *Return*:

   ```
   cd
   ```

   Your current working directory is your home directory.

3. Type the following command and press *Return*:

   ```
   cp dfII_InstallDir/samples/.cdsenv .
   ```

   where *dfII_InstallDir* is the directory path returned by the command you typed on the input line of the CIW in step 1 above.

When you have a valid `.cdsenv` file in your home directory, you can edit the file:

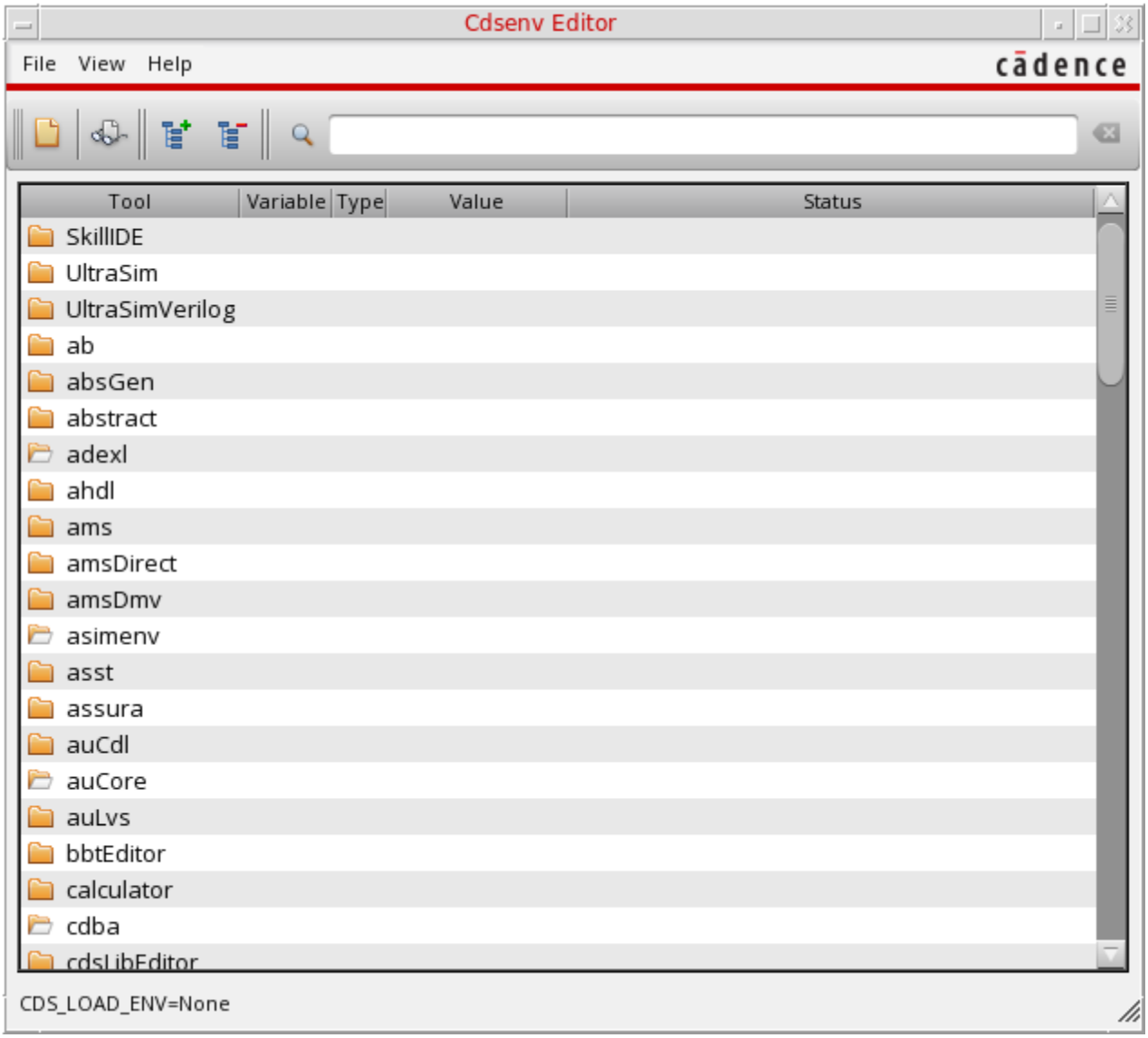
- Using the Cdsenv Editor
- Using a Text Editor

Using the Cdsenv Editor

Cdsenv Editor enables you to search for environment variables across multiple .cdsenv files belonging to different tools. The editor allows you to change values of environment variables and save the changes to the specified file.

To open Cdsenv Editor, in the CIW:

- Select *Options – Cdsenv Editor* or run the `startCdsenvEditor` SKILL function..



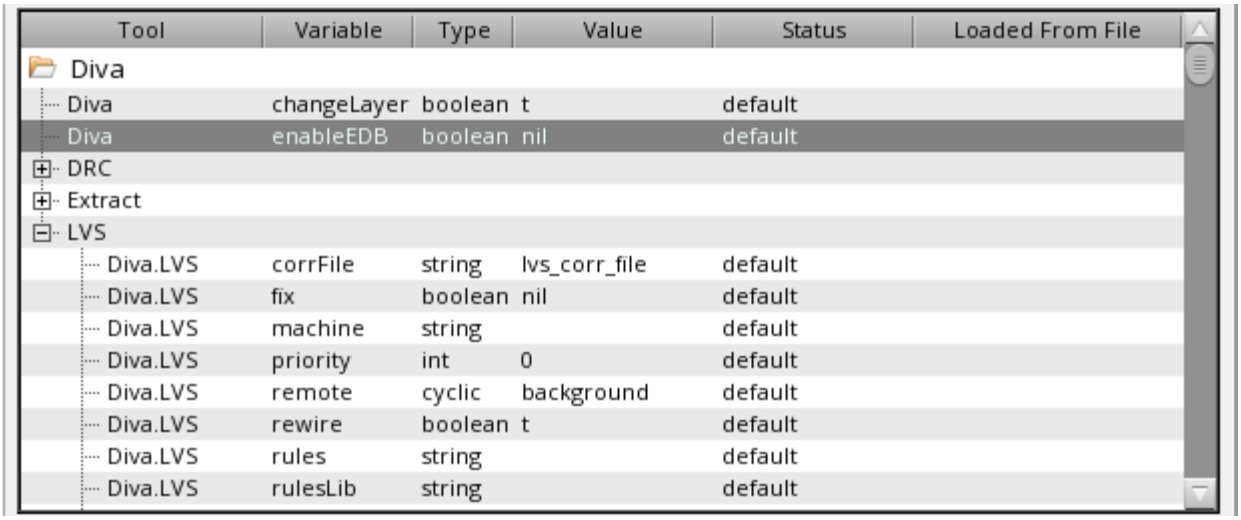
The tools with the open folder icon are loaded into Virtuoso and the tools depicted with closed folder icon are tools are not loaded.

The form contains the following options:

Option	Description
File	Enables you to do the following: ■ ***Save*** Displays the Save .cdsenv file form that allows you to customize how updated environment variables are saved in a .cdsenv file. ■ ***Load File As Overlay*** Displays the Load .cdsenv file dialog box that allows you to load the selected .cdsenv file to overlay the loaded environment variables. ■ ***Quit*** Exits the Cdsenv Editor form.
View	Show/Hide the *Loaded From File* field in the Cdsenv Editor table.
	Displays the Save .cdsenv file form.
	Displays the Load .cdsenv file dialog box that allows you to load the selected .cdsenv file.
	Expands all tools in the Cdsenv Editor table.
	Collapses all tools in the Cdsenv Editor table.
	Searches for the specified environment variable across all .cdsenv files.

Cdsenv Editor Table

The Cdsenv Editor table allows you to navigate to a specific environment variable and update it as needed.



Tool	Variable	Type	Value	Status	Loaded From File
Diva					
Diva	changeLayer	boolean	t	default	
Diva	enableEDB	boolean	nil	default	
DRC					
Extract					
LVS					
Diva.LVS	corrFile	string	lvs_corr_file	default	
Diva.LVS	fix	boolean	nil	default	
Diva.LVS	machine	string		default	
Diva.LVS	priority	int	0	default	
Diva.LVS	remote	cyclic	background	default	
Diva.LVS	rewire	boolean	t	default	
Diva.LVS	rules	string		default	
Diva.LVS	rulesLib	string		default	

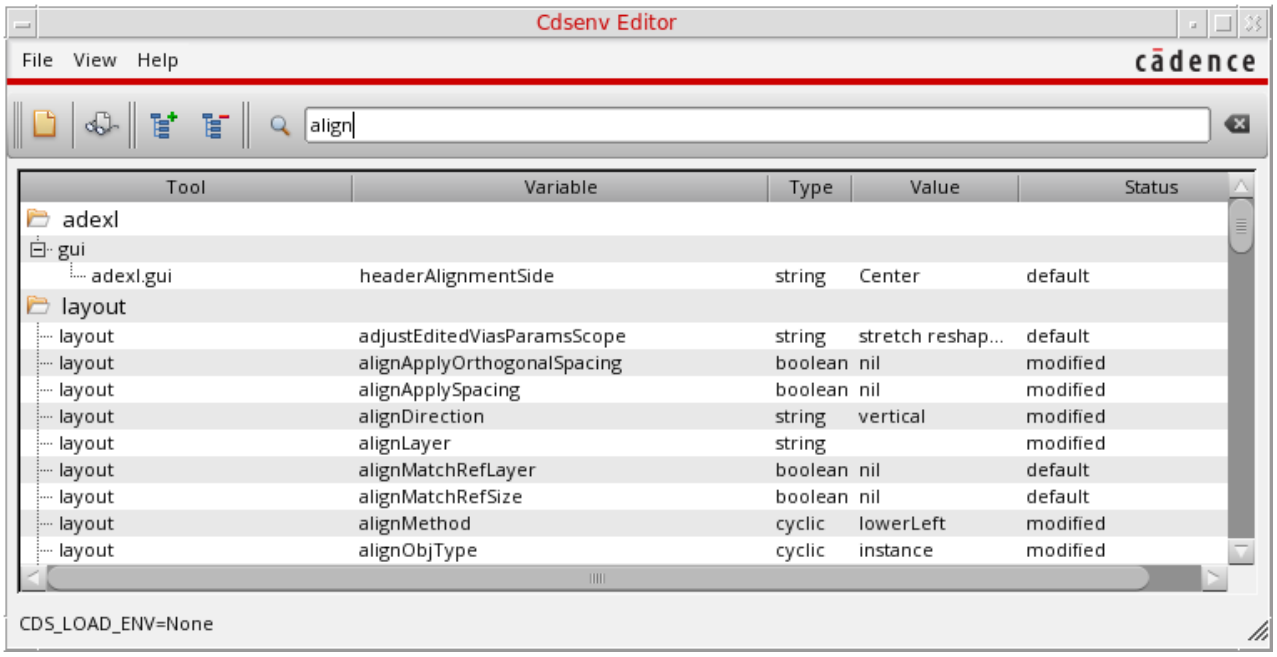
The table comprises the following fields:

- *The Tool* field is an expandable list of tool names that allows you to navigate and display all environment variables for a tool.
- The *Variable* field contains the names of the environment variables.
- The *Type* field specifies the type of environment variables. For example, boolean, string, cyclic, and int.
- The *Value* field specifies the current value of an environment variable. You can edit the specified value of an environment variable from here. To do this, click on the value of the selected variable and then select or type in the value you want to set.
- The *Status* field shows the current status of the variables. If an environment variable has not been edited, the status is displayed as `default`. If you change the value of a variable the status changes to *modified* or *modified from default*. *Modified* means the value has been edited but remains the same as default, while *modified from default* means the value has been changed to a different one from default.
- The *Loaded From File* field is an optional field that shows the file from which the selected .cdsenv file has been loaded. It only shows the path that is not default Cadence installation location.

Editing an Environment Variable in Cdsenv Editor

To edit an environment variable in Cdsenv Editor perform the following steps:

1. Navigate to the environment variable you want to edit in the Cdsenv Editor table or use search to locate it.



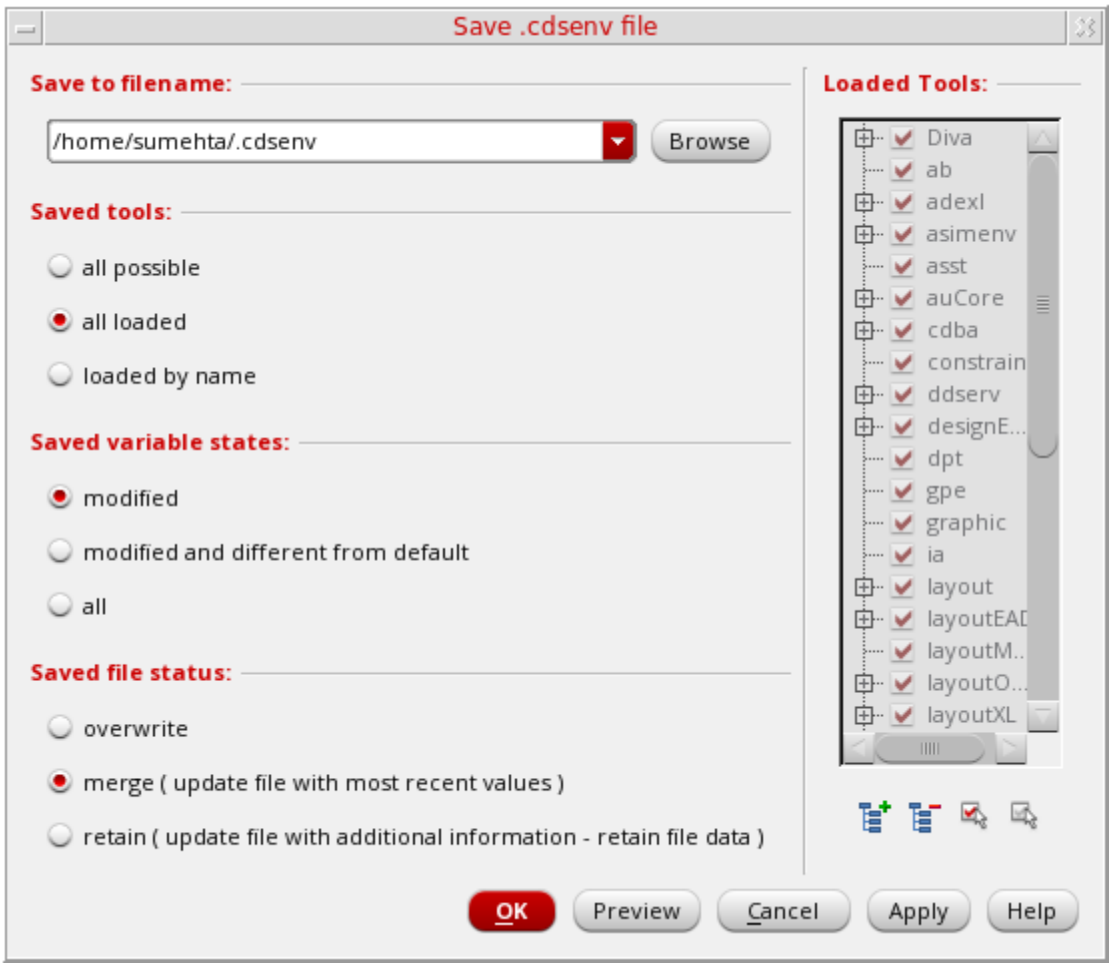
2. Click the value field of the selected environment variable and set the new value.
3. If you change to a non-default value, status is shown as *modified from default*, otherwise the status is *modified*.

To set the value of an environment variable back to its default value, right-click on the variable you want to change and select *Set to default value*.

If an environment variable value is modified in the Cdsenv Editor, the variable row will be highlighted in blue. However, the color stays black if the value is modified using the SKILL command `envSetVal` in the CIW. Environment variable values are synced between Virtuoso and the Cdsenv Editor. Value changes using one tool are instantly reflected on the other tool .

Customizing How Updated Environment Variables are Saved

The Save .cdsenv file form allows you to customize how the environment variables you modify using the Cdsenv Editor are saved.



The form contains the following options:

Option	Description
Save to filename	Specify the .cdsenv file to which you want to save the changes.
Loaded Tools	Displays the tree of tools that have been loaded. The tree becomes available when the *loaded by name* option has been specified under *Saved tools*. You can select the tools you want to save from this tree.
	Expands all tools in the Loaded Tools tree.
	Collapses all tools in the *Loaded Tools* tree.

Option	Description
	Select all tools.
	Deselect all tools.
Saved tools	Enables you to specify the tools for which you want to save the environment variables. You can select: ■ ***all possible*** Saves environment variable updates for all possible tools, includes both loaded and unloaded tools. ■ ***all loaded*** Saves environment variables updates for tools that have been loaded. ■ ***loaded by name*** Saves environment variables updates for only those tools that have been selected in the *Loaded Tools* tree.
Saved variable states	■ ***modified*** Saves only those variables whose values have been edited. ■ ***modified and different from default*** Saves updates to only those environment variables that have been modified and their current value is different from the default value. ■ ***all*** Saves all environment variables in the selected tools.

Option	Description
Saved file status	*Specifies the action taken when an existing .cdsenv file is updated:* ■ ***overwrite*** The existing file is overwritten. ■ ***merge (update file with most recent values)*** Updates are merged in the existing file. Environment variables in the file are updated with the most recent value in the Cdsenv Editor. ■ ***retain (update file* with additional information - retain file data)** Updates are added to the existing file, however, environment variables in the file retain their values.
OK	Saves the changes.
Preview	Displays a dialog box that allows you to preview the updates made. You can view the updates in a tree view or as plain text.
Cancel	Closes the Save .cdsenv file form without saving any changes.
Apply	Saves and applies all changes
Help	Displays form help.

Using a Text Editor

1. Open the `.cdsenv` file with a text editor.

   For example, to use `vi`, type the following command:

   ```
   vi .cdsenv
   ```

2. Make the changes you want (see "Specifying User Interface Preferences in .cdsenv" on page 197).

3. Save your changes and exit the editor.

   Your settings will be loaded into the Cadence environment the next time you start the software.

To apply these settings to the currently running session, you can load the `.cdsenv` file as follows:

➤ On the input line of the CIW, type the following command and press *Return*:

```
envLoadFile("~/.cdsenv")
```

The command and its result (`t` for successful execution) appear in the output area.

See also Chapter 8, "Saving and Recalling Default Settings."

Specifying a Search Order for .cdsenv

You can specify a custom search order for `.cdsenv` by setting the `CDS_LOAD_ENV` environment variable to one of the following values:

Note: For all values except `CSF`, the `CDS_LOAD_ENV` setting applies *after* *your_install_dir*`/tools/dfII/etc/tools/`*binaryName*`/.cdsenv` and *your_install_dir*`/tools/dfII/local/.cdsenv` are loaded.

`False`	Load neither `~/.cdsenv` nor `CWD/.cdsenv`
`CWD`	Load *currentWorkingDirectory*`/.cdsenv` instead of `~/.cdsenv`
`addCWD`	Load `~/.cdsenv`, then load *currentWorkingDirectory*`/.cdsenv`
`CWDElseHome`	Load *currentWorkingDirectory*`/.cdsenv` if it exists, otherwise `~/.cdsenv`, if it exists

`CSF` Load `.cdsenv` files according to the Cadence Setup Search File mechanism

Tip

Specifying `.cdsenv` in your `csfLookupConfig` file is equivalent to setting `CDS_LOAD_ENV` to `CSF`.

The following rules apply:

- The `CDS_LOAD_ENV` setting is not case-sensitive.
- If `CDS_LOAD_ENV` is not set or is not set to one of the above values, the default search order is used (see "Specifying Environment Settings" on page 173).
- The `CDS_LOAD_ENV` setting determines the default save directory (which appears in the *Save To File* field on the Save Defaults form) as follows:

Setting	Default Save To File
`False`	`~/.cdsenv`
`CWD`	*`currentWorkingDirectory`*`/.cdsenv`
`addCWD`	*`currentWorkingDirectory`*`/.cdsenv`
`CWDElseHome`	*`currentWorkingDirectory`*`/.cdsenv` if it exists, else `HOME/.cdsenv`

Specifying User Interface Preferences in .cdsenv

To find the "User Preference Defaults" section in the file, do the following:

1. Open the `.cdsenv` file in a text editor. For example:

   ```
   vi .cdsenv
   ```

2. Find the "User Preference Defaults" sections. For example, in `vi` type

   ```
   /;User
   ```

The user preference defaults have a format like this:

```
ui    winPlaceStyle    boolean    nil
```

See also

- Specifying Window Controls in .cdsenv on page 197
- Specifying Command Controls in .cdsenv on page 200
- Specifying CIW Controls in .cdsenv on page 202
- Specifying Log Filter Options in .cdsenv on page 206
- Specifying CIW Color Choices in .cdsenv on page 207

Specifying Window Controls in .cdsenv

Once you have copied the default `.cdsenv` file from the samples directory (see "Copying and Editing the Default .cdsenv File" on page 187), you can change the default values of the window controls.

To specify default values for window controls in your `.cdsenv` file, do the following:

1. For each control whose default you want to change, edit its value (see the table below).
2. (Optional) Delete settings for those controls whose default values you do not want to change.
3. Save and exit the file.

Note: Alternatively, you can specify most of these settings on the User Preferences form (see "Specifying Window Controls" on page 128).

Default Setting				User Preferences Form Setting
`ui winPlaceStyle`	`boolean`	`nil`		*Place Manually*
`ui setWinTraversal`	`boolean`	`nil`		*Create New Window When Descending*
`ui showScrollBars`	`boolean`	`nil`		*Scroll Bars*
`ui tearOffMenus`	`boolean`	`t`		*Tear-Off Menus*
`ui enableMenuShortcuts`	`boolean`	`t`		*Menu Shortcuts*
`ui showMouseBar`	`cyclic`	`"Bottom"`		*Mouse Prompts*
This setting specifies the location of the mouse bindings line in the window: {`Bottom` | `Top` | `None`}. Select `None` to prevent the mouse bindings line from appearing.				
Note: The `showMouseBar` setting is overridden by any applied workspace regardless of the selected position.				
`ui mouseWheelSpeed`	`int`	`20`		*Mouse Prompts*
This setting modifies speed of the mouse wheel speed. Valid values are between `0` and `1000`. The default is `1000`. The value `0` disables the filtering of events.				
Note: This setting only affects the speed of mouse wheel events processed in the graphics canvas for mouse wheel bindkeys for buttons 4 through 7 (buttons 4 and 5 are forward and backward, 6 and 7 are for mouse wheels that support sideways movement). It does not modify speed of scrolling in the text windows or for other scrollbars, scrollable widgets, or the CIW.				
`ui sessionFocusPolicy cyclic "CanvasDelay"`				*Focus Policy*
`focusPolicy` could either be `"CanvasDelay"`, `"CanvasAsstDelay"`, or `"Click"`				

Default Setting	User Preferences Form Setting
`ui canvasFocusDelay int 200`	*Focus Delay*
`ui sideDockTabs cyclic { "Bottom" | "Side"}` This setting specifies the location of the tabs of dockable assistants in the window: `{Bottom | Side}`.	*Side Dock Tabs*
`ui defaultEditorBackgroundColor string "color"` This setting lets you to change the background color of the editor window. You can reset the background color (`color`) by specifying either the color name (such as "`black`" or "`red`") or its hexadecimal value (such as "`#dcdcdc`" for light gray or "`#cce8c3`" for light green). Default: `black`	*Default Editor Background Color*
`ui defaultDragColor string "color"` This setting changes the highlight color in the graphical editor window (such as, layout and schematics). The highlight color is used for drawing the selection box and zoom rectangle and certain other selection, highlight, and drag operations. Default: `yellow`.	None
`ui enableFileDialogNameCompletion boolean t` This setting controls whether the name auto-complete feature is enabled in the QFileDialog boxes displayed using hiDisplayFileDialog and related APIs.	QFileDialog

Specifying Command Controls in .cdsenv

Once you have copied the default `.cdsenv` file from the samples directory (see "Copying and Editing the Default .cdsenv File" on page 187), you can change the default values of the command controls.

To specify default values for command controls in your `.cdsenv` file, do the following:

1. For each control whose default you want to change, edit its value (see the table below).
2. (Optional) Delete settings for those controls whose default values you do not want to change.
3. Save and exit the file.

Note: You can specify these settings on the User Preferences form (see "Specifying Window Controls" on page 128).

Default Setting	User Preferences Form Setting
`ui infix boolean nil`	*Infix*
`ui showOptionForms boolean nil`	*Options Displayed When Commands Start*
`ui optionFormsStayOnTop boolean nil` When you set this environment variable to `t`, option forms stay on top of all windows, including non-Virtuoso windows, such as terminal windows. To ensure that all option forms follow the same behavior, the environment variable value passed to the first option form is used for all following option forms. The default is `nil`. **Note:** This feature is supported only by Window Managers (WM) that support *Extended Window Manager Hints*. For example, this feature is not supported by mwm.	None

Default Setting	User Preferences Form Setting
`ui undoLevel int 128`	*Undo Limit*
You can also add the following environment variable in the `.cdsenv` file to enable displaying of INFO message for undo/redo changes: `envSetVal("cdba.trace" "dbEnableUndoTrace" 'boolean t)`	
`ui nestLimit int 5`	*Nest Limit*
`ui dblClkTime int 200`	*Double Click Time*
`ui beepVolume int 0`	*Beep Volume*
`ui webBrowser string "firefox"`	*Web Browser*
`ui defaultFloatFieldFormat string "%.6g"`	None
Sets the default floating point format for all hi float fields.	
`ui interruptCheckInterval int 500`	None
Sets the interrupt interval for Ctrl+C. The default value is 500 milliseconds and the range of values can vary from 100 to 1000 milliseconds, that is, from one-tenth second to one second. **Note:** If `Ctrl+C` does not work, `Shift+Ctrl+C` may be used to interrupt the execution of the current command. However, using `Shift+Ctrl+C` is more likely to leave virtuoso in an undetermined state.	

Specifying CIW Controls in .cdsenv

Once you have copied the default `.cdsenv` file from the samples directory (see "Copying and Editing the Default .cdsenv File" on page 187), you can change the default values of the Command Interpreter Window (CIW) controls.

To specify default values for CIW controls in your `.cdsenv` file, do the following:

1. For each control whose default you want to change, edit its value (see the table below).
2. (Optional) Delete settings for those controls whose default values you do not want to change.
3. Save and exit the file.

Note: You can specify most of these settings on the User Preferences form (see "Specifying Window Controls" on page 128).

Default Setting	User Preferences Form Setting
`ui ciwCmdInputLines int 1`	*Input Area Lines*
`ui ciwOutputWrapMode string "Word or Anywhere"` Valid values include: `"Word or Anywhere"` (default), `"None"`, `"Word"`, or `"Anywhere"`.	*Output Wrap Mode*
`ui ciwLogHistorySize int 1000`	*Output Area Lines*
`ui ciwCmdExecuteOnEnter boolean t`	*Enter Key Executes Command*
`ui ciwSyntaxHighlighting boolean t`	*Syntax Highlighting*
`ui ciwTabStop int 4`	*Tab Stop*
`ui ciwRetainUniqueCmds boolean t`	*Retain Unique Commands*
`ui ciwCmdHistoryInPlace boolean t`	*History In Place*

Note: Starting from this release, the `ciwCmdHistorySize` is no longer in use. It is only available for compatibility reasons. Now, the complete history of typed commands is visible at all times.

Default Setting				User Preferences Form Setting
`ui`	`ciwCmdHistorySize`	`int`	`0`	*Input Buffer Lines*
`ui`	`ciwCmdInputLines`	`int`	`1`	*Input Area Lines*
`ui`	`ciwLogHistorySize`	`int`	`1000`	*Output History*
`ui`	`ciwCmdExecuteOnEnter`	`boolean`	`t`	*Enter Key Executes Command*
`ui`	`ciwSyntaxHighlighting`	`boolean`	`t`	None
	This setting controls whether syntax highlighting is on or off. The program reads this setting only when you first start the software: To change this setting you must exit the software, edit your `.cdsenv` file, and restart the software.			

Customizing Titles for Windows and Icons

You can customize the titles for windows and icons in a Virtuoso session in your `.cdsenv` file as shown below. Currently, there is no GUI option for this environment variable.

To customize a window title, add this line to your `.cdsenv` file:

```
designEditor.window windowNameFormat string "winTitle"
```

To customize an icon title, add this line to your `.cdsenv` file:

```
designEditor.window iconNameFormat string "%c"
```

To customize the modified cellview indicator in the title:

```
designEditor.window cellviewModifiedIndicator string
"NewIndicatorCharacter"
```

Valid value is a string that is displayed verbatim in a window title and/or an icon label replaced with the following % constructs:

`%a` Enforces the application name (%a) in a window title.

This means that if the string returned by
`envGetVal( "designEditor.window" "windowNameFormat" 'string)`

does not contain sub-string "`%a`", it is automatically added in the front of the return value.

For example, if a user sets
`envSetVal( "designEditor.window" "windowNameFormat" 'string "%l %c %v")`
a layout session window title will become:
`Virtuoso (R) Layout Suite L <lib-name> <cell-name> <view-name>`

But if it is set to "`%l %c %v, %a`", the window title will read as follows:
`<lib-name> <cell-name> <view-name>, Virtuoso (R) Layout Suite L`

`%m` Specifies the access mode, such as Editing

`%l` Specifies the lib name

`%c` Specifies the cell name

`%v` Specifies the view name

`%x`	Specifies the configuration, such as, Config: `<lib> <cell> <view>` (out of context)

Examples

```
designEditor.window windowNameFormat string "%a %m: %l %c %v  %x"

designEditor.window iconNameFormat string "%c"
```

Specifying Log Filter Options in .cdsenv

Once you have copied the default `.cdsenv` file from the samples directory (see "Copying and Editing the Default .cdsenv File" on page 187), you can change the default values of the log file filter options.

To specify default values for log file filter options in your `.cdsenv` file, do the following:

1. For each control whose default you want to change, edit its value (see the table below).
2. (Optional) Delete settings for those controls whose default values you do not want to change.
3. Save and exit the file.

Note: You can specify most of these settings on the Set Log File Display Filter form (see "Changing the Log Filter Options" on page 148).

Default Setting				Form Setting
`ui`	`typedReturnValue`	`boolean`	`t`	*Return values from typed-in commands*
`ui`	`errorOutput`	`boolean`	`t`	*Error message output*
`ui`	`warningOutput`	`boolean`	`t`	*Warning message output*
`ui`	`standardOutput`	`boolean`	`t`	*Standard message output*
`ui`	`accelInput`	`boolean`	`nil`	*Accelerated input*
`ui`	`accelReturnValue`	`boolean`	`nil`	*Return values from accelerated input*
`ui`	`promptOutput`	`boolean`	`nil`	*Prompt output*

Specifying CIW Color Choices in .cdsenv

Important

These color specifications are loaded only during startup and cannot be changed interactively. The software uses a 24-bit TrueColor visual by default. If it cannot find it, it looks for a 16-bit TrueColor visual. If it cannot find that either, it looks for a 15-bit TrueColor visual. If it cannot find any of these, the software will not run.

You can specify the colors used in the Command Interpreter Window (CIW) when displaying the following items:

- Error and warning messages that appear in the output area
- Parentheses-matching and command-matching highlighting on the input line

To specify colors used in the CIW, do the following:

1. Add one of the following lines to your `.cdsenv` file for each color you want to specify:
   a. To specify the color for warning messages that appear in the CIW output area, add
      `ui ciwWarnColor string "color"`
   b. To specify the color for error messages that appear in the CIW output area, add
      `ui ciwErrorColor string "color"`
   c. To specify the color for parentheses-matching on the input line of the CIW, add
      `ui ciwMatchParenColor string "color"`
   d. To specify the color for parentheses-mismatching on the input line of the CIW, add
      `ui ciwMismatchParenColor string "color"`
   e. To specify the color for command-matching on the input line of the CIW, add
      `ui ciwMatchCmdColor string "color"`
2. Save and exit the file.

You can specify a color (*color*) using either its X color name (such as "orange" or "red") or its encoded RGB name (such as "#dcdcdc" for light gray or "#cce8c3" for light green).

Tip

You can change the background of the CIW to gray by setting the `CDS_GRAY_BG` environment variable (`setenv CDS_GRAY_BG`) before starting any tools which use the Cadence style. To switch the gray background off, use `unsetenv CDS_GRAY_BG`.

Specifying Memory Check Intervals in .cdsenv

The following environment variables are used to control the frequency of memory check:

- `ui memoryCheckIntervalSeconds int 1`

  You can set the memory check interval from 0 through 60. If the memory check interval is set to 0 then there will be no memory checking and if it is set from 1 to 60 then that will be the number of seconds between each check.

  Default value is 1.

- `ui systemMemoryCheckInterval int 10`

  You can set the system memory check interval from 1 to 100.

  Default value is 10.

  The idle memory check is calculated on the basis of `memoryCheckIntervalSeconds` * `systemMemoryCheckInterval`. For example, if `memoryCheckIntervalSeconds` is set to 5 and `systemMemoryCheckInterval` is set to 4, then the idle memory check timer will run every 20 seconds.

Controlling Stroke Text in the Canvas

The following environment variable controls whether or not stroke text in the canvas is drawn with the defined `lineStyle` for the pen assigned to the text.

```
ui     useLineStyleForStrokeText     boolean     nil
```

If the environment variable is `nil` (the default value), stroke text is drawn with solid lines irrespective of the `lineStyle` setting.

Auto-Selecting the Content

When the Tab key is pressed to traverse through the editable string fields, the following environment variable (if set to `true`) auto selects the content in the text field:

```
ui     focusToFieldSelectsText     boolean     t
```

Determining the Width of Bottom Assistants

The following environment variable determines the width of the bottom assistant in a window.

```
ui    bottomAsstSpansFullWidth    boolean    t|nil
```

If set to `t`, the assistant at the bottom will be displayed along the entire bottom area of the window. If set to `nil`, the assistant at the bottom will lie between the left and right assistants.

The default value is `nil`.

Specifying Default Application in .cdsenv

You can specify the default application for schematic, symbol, and layout editors in your `.cdsenv` file as shown below.

See Setting the Default Application for a Cellview for information on how to specify the default application in the Virtuoso Design Environment.

Note: You can also specify a default application *level* for the schematic and symbol editors, but the default application setting overrides the default application level setting.

To specify the default application for the schematic editor, add this line to your `.cdsenv` file:

```
graphic schematicDefaultApp string "appString"
```

Valid values for *appString* are as follows:

```
Schematics L
Schematics XL
ADE Explorer
ADE Assembler
```

To specify the default application for the symbol editor, add this line to your `.cdsenv` file:

```
graphic schematicSymbolDefaultApp string "appString"
```

Valid values for *appString* are as follows:

```
Symbol L
Symbol XL
```

To specify the default application for the layout editor, add this line to your `.cdsenv` file:

```
graphic maskLayoutDefaultApp string "appString"
```

Valid values for *appString* are as follows:

```
Layout XL
Layout EXL
```

To specify the default level for the schematic and symbol editor applications, add this line to your `.cdsenv` file:

```
graphic schematicDefaultTier string "appLevel"
```

Valid values for *appLevel* are as follows:

`L`	for Schematics L and Symbol Editor L
`XL`	for Schematics XL and Symbol Editor XL

After you save and exit the file, the program loads these settings the next time you run your Cadence software.

See also "Using Different Application Tiers" on page 62.

Specifying License Checkout Behavior in .cdsenv

Applications, such as Virtuoso Analog Design Environment (ADE), Virtuoso Schematic Editor (VSE), and Virtuoso Layout Suite (VLS) have tiered functionality levels. See Using Different Application Tiers.

You can specify the following environment variables in the `.cdsenv`, file to specify the order of checkout of available licenses.

- `VLSLicenseCheckoutOrder`
- `VSELicenseCheckoutOrder`
- `VLSAdvOptLicenseCheckoutOrder`
- `EADLicenseCheckoutOrder`
- `maestroCheckoutOrder`
- `VIVALicenseCheckoutOrder`

For information on specifying these environment variables, see CheckoutOrder Variables in *Virtuoso Software Licensing and Configuration User Guide*.

Use Next License Variables

The following `.cdsenv` variables can also be set to specify how to use the next available product license:

- `VSEL_UseNextLicense`
- `VLSL_UseNextLicense`
- `VLSXL_UseNextLicense`

These environment variables can have the following values set:

Value	Use
`prompt`	If set, the Next License dialog is displayed each time a requested license is not available. **Note:** During a Virtuoso session, the Next License dialog will only display once for each application.
`always`	If set, Virtuoso will always try to check out another license if the requested license is not available.
`never`	If set, Virtuoso will never attempt to check out another license.

Note: The default setting for all of these variables is `prompt`.

For more information, see *UseNextLicense Variables* in the *Licensing Environment Variables* chapter and *Use Next License Dialog Box* in the *Configuring the Virtuoso Design Environment* chapter of *Virtuoso Software Licensing and Configuration User Guide*.

When `prompt` is set as the current value, the Next License dialog box is displayed asking you to select the next license in the specified checkout order list.

Here are the options that you can select:

Option	Action when selected
Session	Virtuoso attempts to check out another license. If the requested license is not available. It is applied inside of the current session. The Next License dialog box is displayed again when Virtuoso is restarted.
Skip	Skips the next license and tries to check out next license in check out order.
Always	Virtuoso always attempts to check out another license and the appropriate next license environment variable (for example, `VLSL_UseNextLicense`) is updated in the `.cdsenv` (to `always`). The Next License dialog box is not displayed again when Virtuoso is restarted.
Never	Virtuoso does not try to check out another license and the appropriate next license environment variable (for example, `VLSL_UseNextLicense`) is updated in the `.cdsenv` (to `never`). The Next License dialog box is not displayed again when Virtuoso is restarted.

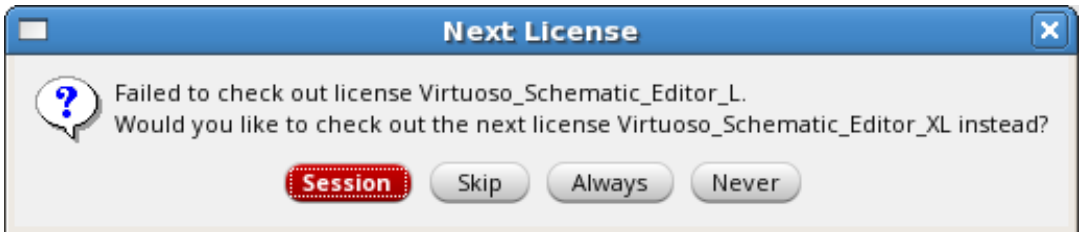


Figure 7-2 CIW Excerpt: VLSL_UseNextLicense Set to Prompt

8

Saving and Recalling Default Settings

Cadence software comes with factory pre-set default settings so that you can run the software immediately. When forms appear, the default values are already set. Some buttons are already checked/unchecked, some fields already contain text, and some other choices are already set.

Cadence supplies the following sample file containing default settings:

your_install_dir/tools/dfII/samples/.cdsenv

Your system administrator customizes this file for your site and puts it in the following local environment file:

your_install_dir/tools/dfII/local/.cdsenv

You can override these site-specific settings by creating another `.cdsenv` file in your home or workarea directory. When the Cadence software loads, it reads your `.cdsenv` file after the site-specific file, so the settings in your file override the settings in files loaded earlier. These overrides apply to your system only.

Note: Executable SKILL commands in your `.cdsinit` file will take precedence over settings in your `.cdsenv` file if you have both files in your home or workarea directory.

The following topics are discussed:

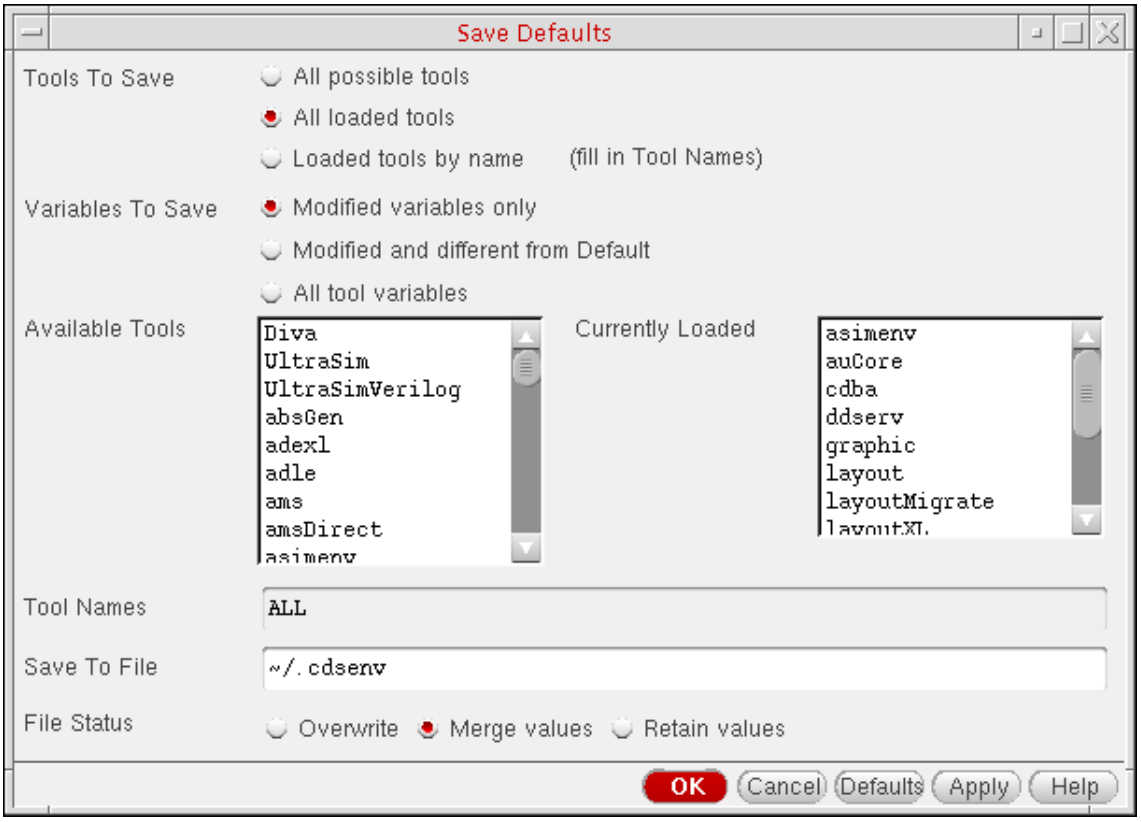
See also "Copying and Editing the Default .cdsenv File" on page 187 and "Specifying a Search Order for .cdsenv" on page 195.

Specifying New Default Values

To specify and save new default values from one session to the next, do the following:

1. From the Command Interpreter Window (CIW), choose *Options – Save Defaults*.

   The Save Defaults form appears.



2. Select one of the following *Tools To Save* radio buttons to specify the environments to save:

   - *All possible tools* saves changed settings for all applications listed in the *Available Tools* list area.

     The *Variables To Save* radio buttons become inactive.

   - *All loaded tools* (the default) saves changed settings for all applications listed in the *Currently Loaded* list area.

   - *Loaded tools by name* saves changed settings for individual applications whose names you type in the *Tool Names* field (such as `schematic`).

The *Tool Names* field becomes available when you select the *Loaded tools by name* radio button. The default value in that field is `ALL` (to save settings for all loaded tools).

3. Select one of the following *Variables To Save* radio buttons.

   Note: These radio buttons are not available if you selected *All possible tools* above.

   - *Modified variables only* saves only those variables that you modified.
   - *Modified and different from Default* saves those variables that you modified and that changed from their default values.
   - *All tool variables* saves all application variables.

4. (Optional) In the *Save To File* field, change the name and location of the environment file to save.

   The default is the `.cdsenv` file in your home directory (`~/.cdsenv`).

5. Select one of the following *File Status* radio buttons:

   - *Overwrite* overwrites matching settings
   - *Merge values* merges changed settings in with existing settings
   - *Retain values* retains existing settings and will not overwrite matching values

6. Click *OK*.

Settings are saved to the `.cdsenv` file and read the next time you start the software.

See also "Recalling Defaults".

Note: Some applications require you to save application-specific defaults in a file other than `~/.cdsenv`. Those applications provide a command to load the defaults. For that information, see the documentation for the application.

Recalling Defaults

To load a saved default file for your design session:

➤ On the CIW input line, type the following command:

```
envLoadFile("~/.cdsenv")
```

Environment settings from this file are loaded for this design session.

9

Getting Started with Workspaces

A *workspace* is a layout of configurable user interface components that you can customize to assist your work in a particular application or when you perform a particular task. You can use a workspace to define the existence, size, and position of a set of associated user interface components within an application.

Toolbars and assistant panes are the two main user-interface components you can configure in your workspace. The toolbars and assistant panes you can request as part of your workspace depend on the currently active application/view type.

You can choose a particular arrangement of *assistant panes* (dockable workspace components that can be resized and repositioned in a session window) and toolbars available to the current application.

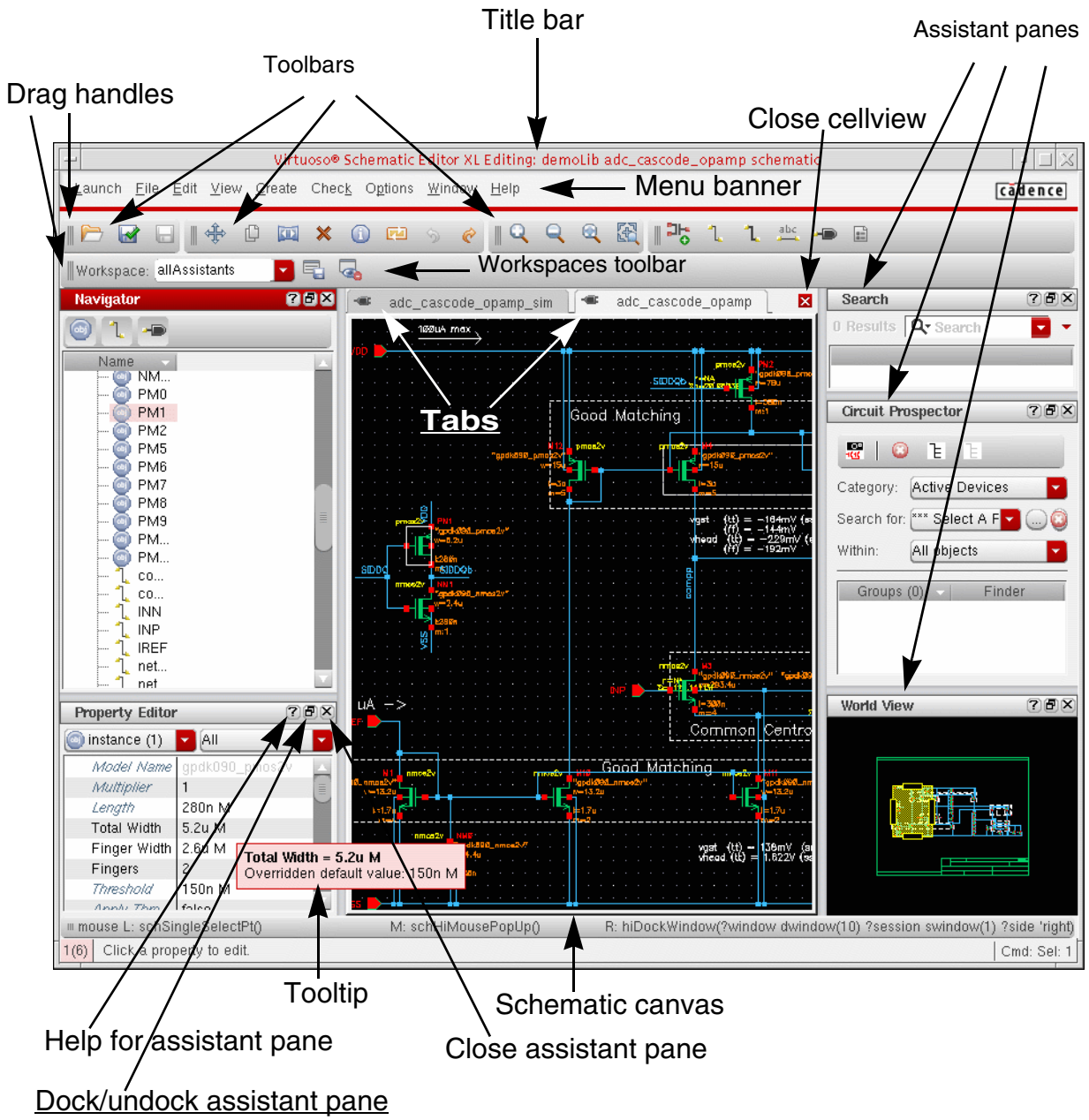
Cadence provides a set of workspaces to get you started (see Chapter 10, "Working with Workspaces"). In addition, circuit designers, layout engineers, and members of your company's CAD group can define custom workspaces that specify what assistant panes and toolbars appear, and under what circumstances .

Note: You can find Cadence-provided `.workspace` files and `workspace.default` files in *`your_install_dir`*`/share/cdssetup/dfII/workspaces/`*`appDirs`*.
Each `.workspace` file is a binary file that contains information defining the workspace configuration. The `workspace.default` file in each application directory (*`appDir`*) is an ASCII text file that contains the name of the default workspace for that application.

See the following sections for more information:

Sample Session Window and Workspace

Here is a sample workspace showing the various elements of a workspace in a Virtuoso Schematic Editor XL Editing session window. A session window appears when you first open a cellview for design editing.



What You Can Configure

You can define the following properties of your window layout in a workspace:

- What assistant panes are docked, floating, and hidden
- What toolbars are docked and hidden
- Positions and geometries of the assistant panes and toolbars in your window layout

You can specify only those user interface components that are available for a particular viewtype or application as part of your workspace. For example, you cannot create a workspace in the Virtuoso Layout XL that contains an assistant pane that is specific to the ADE Explorer environment.

You cannot define what banner menus appear, what icons appear on a toolbar, or the content of an assistant pane using a workspace.

You can customize a workspace to appear for a particular view type or application (see also "Setting the Default Workspace for an Application" on page 251). For example, you can specify one combination of toolbars, assistant panes, and so on, to appear for a schematic view and a different combination for a layout view.

Note: A workspace applies to a single session window (rather than to a single UNIX process). Therefore, changing the workspace in one schematic session window does not affect other schematic session windows. For more information see Session Windows and Workspaces.

Workspace Search Order

When you create a custom workspace, whether for your own use or for your site, the built-in Cadence search mechanism determines which configuration appears in a particular situation. CAD staff, project managers, and end-users can define what they consider to be appropriate configurations for a given task or application. Your locally defined workspaces take precedence over any others. The default search order is as follows:

1. Your locally defined workspaces
2. Workspaces defined for your site
3. Cadence workspaces

See Chapter 10, "Working with Workspaces"

You can change the search order by editing the `setup.loc` file. (You can find a sample `setup.loc` file in *your_install_dir*`/share/cdssetup`.) The `setup.loc` setup file controls the search order the program uses to determine what workspace should appear. The program looks for workspace files that are stored as follows using the search order specified in `setup.loc`:

`.cadence/dfII/workspaces/`*appNameDir*`/`*workspaceName*`.workspace`

where *appNameDir* might be any of the following:

```
Layout
Schematics
Schematics_XL
Symbol
Symbol_XL
VLS-EXL
VLS-EXL_Schematics
Virtuoso_XL
Virtuoso_XL_Schematics
adegxl-schematic
adexl-schematic
```

For more information about the `setup.loc` file, see the Cadence Setup Search File: setup.loc in *Cadence Application Infrastructure User Guide*.

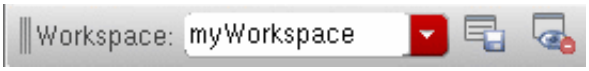
The program uses the first matching workspace definition it finds. Virtuoso Design Environment applications use the following algorithm to determine the set of available workspaces for each application:

- For each registered tool `$t` in each registered application (such as `adexl-schematic`)
  - For each search path component `$c` in the `setup.loc` file
    - For each matching file `$f` in `$c/.cadence/dfII/workspaces/$t/*.workspace`

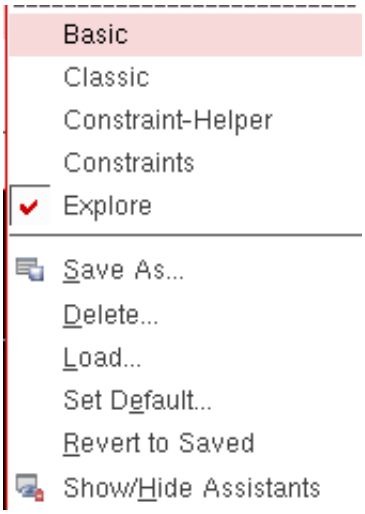
Cache the definition of workspace `$f` for tool `$t`

Note: The `.cadence` directory is not part of the path for workspaces shipped with your Cadence software: *`your_install_dir`*`/share/cdssetup/dfII/workspaces/$t`.

The program lists cached workspaces in the drop-down combo box on the Workspaces toolbar



and on the *Window – Workspaces* submenu.



Consider a setup file stored as *`your_install_dir`*`/share/local/cdssetup/setup.loc` and containing the following :

```
.
$HOME
your_install_dir/share/local/cdssetup
$ (compute:THIS_TOOL_INST_ROOT)/share
```

For example, consider the following set of available *myXLconstraints* workspace files:

```
/home/user/.cadence/dfII/workspaces/Virtuoso_XL/myXLconstraints.workspace
/home/user/project/bigchip/.cadence/dfII/workspaces/Virtuoso_XL/
myXLconstraints.workspace
```

If you start Virtuoso Layout Suite XL from `/home/user/project/bigchip`, the program finds `myVXLconstraints.workspace` in the following location first:

```
/home/user/project/bigchip/.cadence/dfII/workspaces/Virtuoso_XL/
```

If you start Virtuoso Layout Suite XL from any other location, the program finds `myVXLconstraints.workspace` in the following location:

```
/home/user/.cadence/dfII/workspaces/Virtuoso_XL/
```

Additional notes about workspaces in the context of layout and schematics:

- If you open a schematic from Virtuoso Layout Suite XL, the program searches `.cadence/dfII/workspaces/Virtuoso_XL_Schematics` for workspace files.
- If you open a schematic from Virtuoso Layout Suite EXL, the program searches `.cadence/dfII/workspaces/Virtuoso_GXL_Schematics` for workspace files.
- If you open a schematic from anywhere else, the program searches `.cadence/dfII/workspaces/Schematics` or `.cadence/dfII/workspaces/VSE-XL` for workspace files, depending on which program you launch.

Consider the following set of *basicXL* workspace files:

```
/home/user/.cadence/dfII/workspaces/Virtuoso_XL/basicXL.workspace
/home/user/.cadence/dfII/workspaces/Virtuoso_XL_Schematics/basicXL.workspace
your_install_dir/share/local/cdssetup/.cadence/dfII/workspaces/Virtuoso_XL/
basicXL.workspace
$(compute:THIS_TOOL_INST_ROOT)/share/cdssetup/dfII/workspaces/
Virtuoso_XL_Schematics/basicXL.workspace
```

If you start Virtuoso Layout Suite XL from `/home/user/project/bigchip`, the program finds `basicXL.workspace` in the following location first:

```
/home/user/.cadence/dfII/workspaces/Virtuoso_XL/
```

If you open a schematic at this point, the program finds `basicXL.workspace` in the following location:

```
/home/user/.cadence/dfII/workspaces/Virtuoso_XL_Schematics/
```

See also "Using Different Application Tiers" on page 62 for application licensing information.

Session Windows and Workspaces

A session window appears when you first open a cellview for design editing. Each session window maintains its own set of workspaces. As such, you can have two session windows of the schematic editor that have two different workspaces. You might have one session window that is a schematic view with one workspace while another session window is a schematic view with a different workspace.

Note: You can switch between session windows using *Alt+Tab*.

Also, changing the selected workspace in one session window does not affect the workspace in another session window, even if the same cellview appears in both session windows.

Note: See Chapter 10, "Working with Workspaces" for more information about the *Classic* and *ISearch* workspaces.

Tabs in a Session Window

You can have one or more cellviews open on one or more tabs in a session window (see "Opening a Cellview" on page 104). For example, you might have three tabs on which you have the schematic and symbol views of your `NAND2` cell and the layout view of your `AMPLIFIER` cell open in the same session window. You select which cellview appears in the foreground of your session window by clicking on the corresponding tab header.

Note: See also "Closing a Cellview on a Tab in a Session Window" on page 114.

When you select different tabs in your session window to toggle between different cellviews of the same cell or other cells, the workspace changes to present window components that are relevant to the current view type. Only those menus, toolbars, and assistant panes that are relevant to the application corresponding to the current tab appear.

For example:

- If you are currently using the schematic editor, you can switch from one schematic view to another and the same workspace applies: You see the same assistant panes and toolbars. The content of the assistant panes reflects the schematic cellview you are viewing.
- If you switch from a schematic cellview tab to a layout cellview tab, the workspace for the layout application appears in the session window.

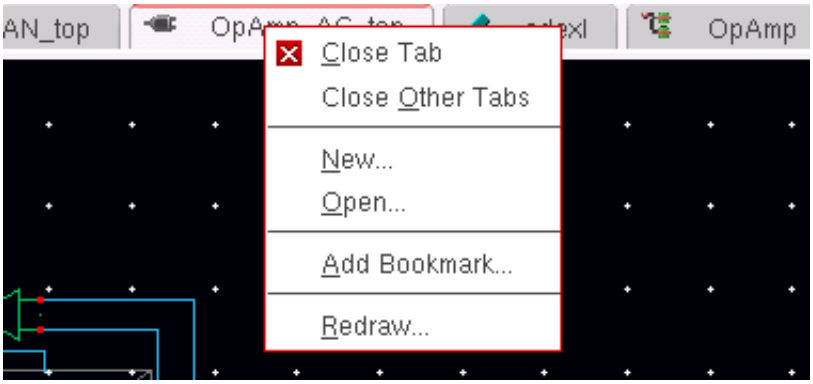
You can have several tabs in your session window, each running a different application. Only one tab is considered to be active at any one time.

When a session window does have multiple tabs, the tab order can be changed by dragging and dropping a particular tab into a new position, moving it from left-to-right or right-to-left as required.

Note: You can switch between tabs using the `Ctrl+Tab` keybinding

Tab Pop-Up Menu Options

If you right click over a tab it will display a pop-up (context) menu.



Option	Description
Close Tab	Closes the current tab.
Close Other Tabs	Closes all other open tabs except for the current tab.
New	Displays the New File Form (see also "Creating a New Cellview" in the *Cadence Library Manager User Guide*).
Open	Displays the Open Cell form (see Opening a Cellview).
Add Bookmark	Displays the Add Bookmark form (see Bookmarking Designs).
Redraw	Refreshes the current design canvas window.

Hierarchy Changes and Workspaces

Your workspace might change if you change from one cellview to another by moving up or down the current hierarchy, even though you are not changing tabs in your session window.

For example, if you descend into a layout view from a schematic view, the program displays the workspace defined for the layout application. Also, if you descend into a symbol cellview from a schematic cellview to perform an edit-in-place operation, the program displays the workspace defined for the symbol editor application.

Note: See also "Tabs in a Session Window" on page 225 and "Setting the Default Workspace for an Application" on page 251 for additional information.

10

Working with Workspaces

Cadence provides a set of workspaces designed to help you perform a set of related tasks, such as working with constraints or performing interactive searches within your design. Each workspace is tailored to a particular application, cellview, or work objective.

As well as being able to select from a set of Cadence-supplied workspaces, you can create your own workspaces that consist only of the user-interface components you want to assist you when working with a particular application, subapplication, or view type.

You can customize a workspace for a particular work objective. For example, you can create a custom configuration for schematic setup (such as `schematicSetup`) and another custom configuration for schematic modifications (such as `schematicMod`).

You can use a workspace to change the way you view your data by choosing what assistant panes and toolbars appear in your window, where they appear, and whether each assistant pane is docked or floating. You cannot use a workspace to manipulate (filter or reduce) the underlying data itself. For example, you cannot use a workspace to restrict which parameters appear on the Property Inspector assistant pane or to program which label display objects appear on the main schematic canvas.

You cannot remove or edit any of the following user-interface components as part of your custom workspace:

- Main menu bar (menu banner) appearing at the top of a design session window

  You cannot alter the existence, location, or content of the main menu bar.

- Main menu bar submenu content

  You cannot remove any menu items from the main menu bar submenus or exchange them between menus.

- Toolbar buttons/icons

  You cannot add or remove tool buttons from a toolbar or swap them between toolbars.

- Context-sensitive menus: right-click menus

  You cannot change the content of these menus.

- Menu bars, toolbars, context-sensitive menus associated with an assistant pane

  You cannot modify any of these additional user-interface components that might appear as part of an assistant pane in your workspace.

Tip

You can save time creating your setup preferences by selecting a Cadence workspace or other custom configuration that closely matches your requirements as a starting point. For more information, see Customizing a Cadence Workspace.

See the following sections for more information:

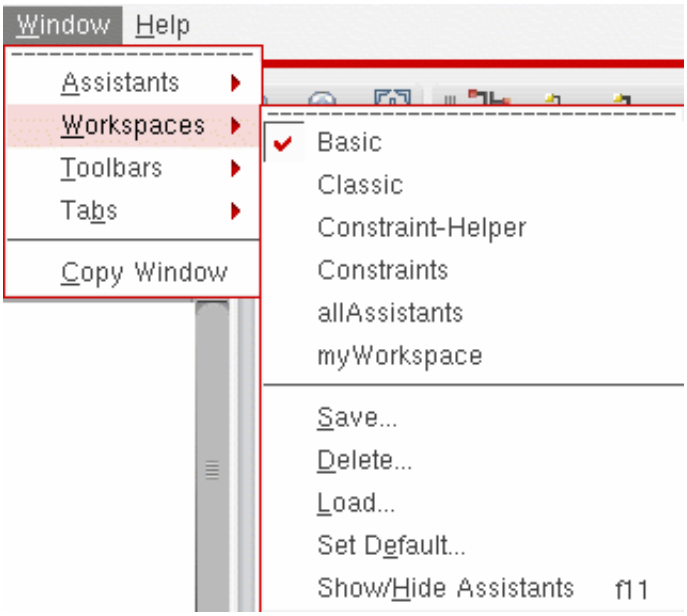
- Selecting a Workspace on page 231
- Adding Assistant Panes and Toolbars to a Workspace on page 232
- Removing Items from Your Workspace on page 234
- Docking or Floating an Assistant Pane on page 237
- Getting Help on an Assistant Pane on page 236
- Hiding Assistant Panes on page 240
- Modifying the Size or Location of an Assistant Pane on page 241
- Modifying the Length or Location of a Toolbar on page 243
- Saving a Workspace on page 244
- Loading a Custom Workspace on page 246
- Deleting a Custom Workspace on page 248
- Customizing a Cadence Workspace on page 249
- Reverting to a Saved Workspace on page 250
- Setting the Default Workspace for an Application on page 251

Selecting a Workspace

To select a workspace using the menu banner, do the following:

1. Choose *Window – Workspaces*.

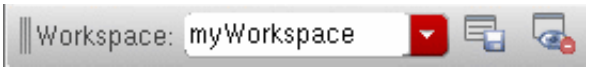
   A submenu of workspaces appears listing only those configurations available for use with the current cellview/application.



2. Select the workspace you want to apply to the current session window.

   The program applies the workspace you selected to the current session window.

Alternatively, you can select a workspace from the drop-down combo box on the Workspaces toolbar.



Assistant panes that are part of a Cadence workspace are initially docked. You can modify the arrangement of your session window and save it as a custom workspace (see "Saving a Workspace" on page 244).

Adding Assistant Panes and Toolbars to a Workspace

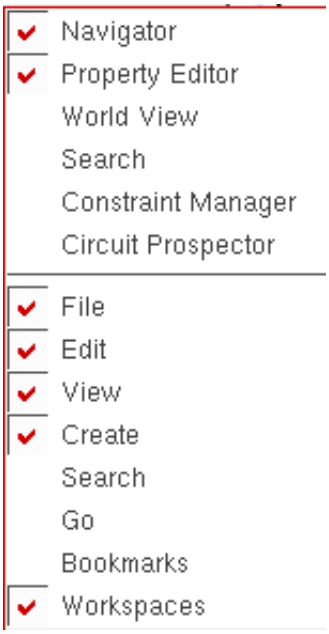
You can add assistant panes and toolbars to your workspace using the *Window* menu or a right-click menu as described below.

Adding a Toolbar or an Assistant Pane Using the Right-Click Menu

To add an assistant pane or toolbar to your custom workspace using the right-click menu, do the following:

1. Right-click the title bar of an assistant pane or a toolbar.

   A menu of those assistant panes and toolbars available to the current application/view type appears. Active items appear with a check mark to the left of their names.



2. Select an item that is not currently active (does not have a check mark to the left of its name).

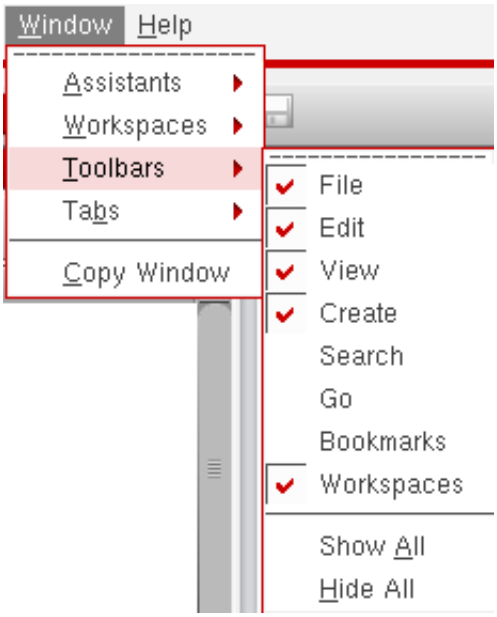
   The item appears in your workspace. If you display the right-click menu again, you will see a check mark next to the item.

Adding a Toolbar Using the Window Menu

To add a toolbar to your custom workspace using the *Window* menu, do the following:

1. Choose *Window – Toolbars*.

   A submenu of toolbars you can configure for the current application/view type appears. Active toolbars appear with a check mark to the left of their names.



2. Select a toolbar name that is not currently active.

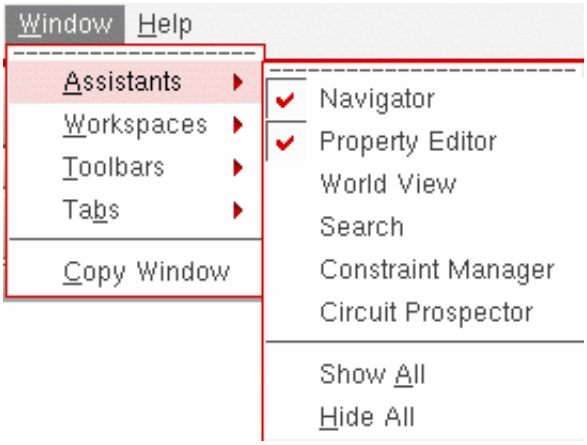
   The toolbar appears in your workspace. If you display the submenu again, you will see a check mark next to the toolbar name.

Adding an Assistant Pane Using the Window Menu

To add an assistant pane to your custom workspace using the *Window* menu, do the following:

1. Choose *Window – Assistants*.

A submenu of assistant panes you can configure for the current application/view type appears. Active panes appear with a check mark to the left of their names.



Note: The *Assistants* menu entry is not available in the Virtuoso Schematic Editor L.

2. Select an assistant pane that is not currently active.

   The pane appears in your workspace. If you display the submenu again, you will see a check mark next to this pane's name.

Removing Items from Your Workspace

You can remove assistant panes and toolbars from your workspace using the *Window* menu or a right-click menu as described below.

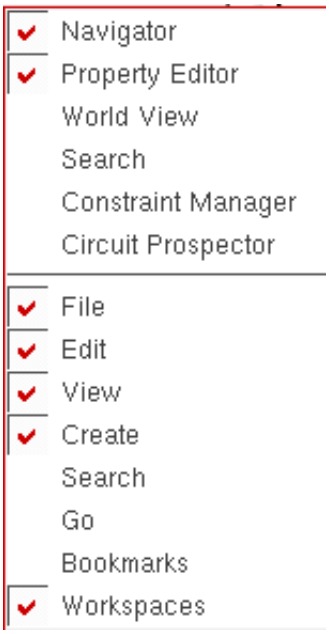
- Removing a Toolbar or an Assistant Pane Using the Right-Click Menu on page 234
- Removing a Toolbar Using the Window Menu on page 235
- Removing an Assistant Pane Using the Window Menu on page 236

Removing a Toolbar or an Assistant Pane Using the Right-Click Menu

To remove an assistant pane or a toolbar from your custom workspace using the right-click menu, do the following:

1. Right-click the title bar of an assistant pane or a toolbar.

A menu of those assistant panes and toolbars available to the current application/view type appears. Active items appear with a check mark to the left of their names.



2. Select an active item to remove the check mark.

   The item disappears from the workspace. If you display the right-click menu again, the check mark no longer appears next to the item.

Removing a Toolbar Using the Window Menu

To remove a toolbar from your custom workspace using the *Window* menu, do the following:

1. Choose *Window – Toolbars*.

   A submenu of toolbars you can configure for the current application/view type appears. Active toolbars appear with a check mark to the left of their names.

2. Select the toolbar you want to remove.

   The toolbar disappears from your workspace. If you display the submenu again, the check mark no longer appears next to the toolbar name.

Removing an Assistant Pane Using the Window Menu

To remove an assistant pane from your custom workspace using the *Window* menu, do the following:

1. Choose *Window – Assistants*.

   A submenu of assistant panes you can configure for the current application/view type appears. Active panes appear with a check mark to the left of their names.

2. Select the assistant pane you want to remove.

   The pane disappears from your workspace. If you display the submenu again, the check mark no longer appears next to this pane's name.

Getting Help on an Assistant Pane

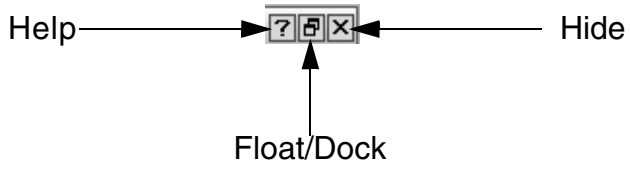
To get help on an assistant pane, do the following:

➤ In the upper right corner of the pane, click the question mark icon.

The Help page for the assistant appears in the Cadence Help window.

Docking or Floating an Assistant Pane

You can make a docked assistant pane float or dock a floating assistant pane using the Float/Dock button in the upper right corner of each pane. The Float/Dock button appears between the Help (**?**) and Hide (**x**) buttons.



To make a docked assistant pane float, do one of the following:

- In the upper right corner of the docked pane, click the Float button.
- In the title bar of the docked pane, double-click.
- Drag a docked pane by its title bar to an open area and release the mouse button.

  The pane is now floating.

To dock a floating assistant pane, do one of the following:

- In the upper right corner of the floating pane, click the Dock button.
- In the title bar of the floating pane, double-click.
- Drag the floating pane close to any edge of your session window until you see its outline where you want it and drop it.

  The pane is now docked.

See also "Modifying the Size or Location of an Assistant Pane" on page 241 for additional information.

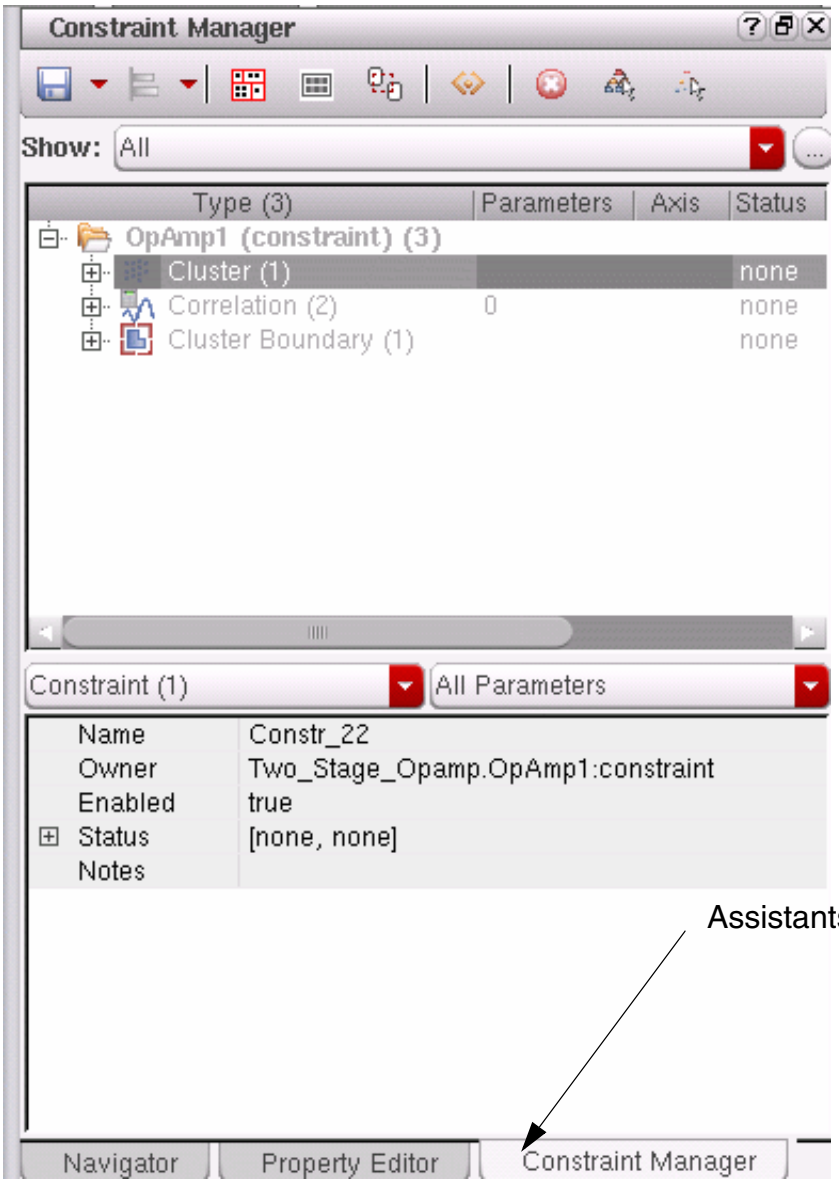
Displaying Assistant Panes as Tabs

You can display assistant panes as tabs by dragging and dropping one assistant on top of another. This can be useful if you have a busy workspace and need to maximize space.

To display an assistant in a tabbed format:

➤ Click and drag the title bar of one assistant and drop it on the body area of another assistant.

Note: You can click the *Float* button (see Docking or Floating an Assistant Pane) to detach an assistant from an existing group of tabbed assistants. Alternatively, you can drag the assistant away from the grouped assistants.



Assistants displayed as tabs

Hiding Assistant Panes

To hide an assistant pane, do the following:

➤ Click the Hide (**x**) button in the upper right corner of the pane.

The program removes the pane from your workspace.

Note: You can add the assistant pane back to your workspace using the *Window* menu or a right-click menu as described in "Adding Assistant Panes and Toolbars to a Workspace" on page 232.

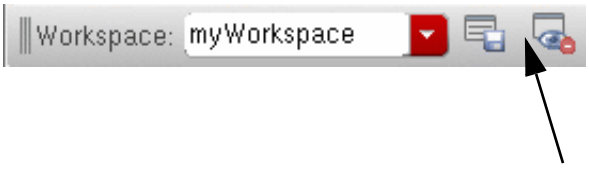
See also the following topics:

- Hiding All Assistant Panes with a Single Action on page 240
- Toggling Assistant Panes on page 241

Hiding All Assistant Panes with a Single Action

To hide all assistant panes with a single action, do one of the following:

➤ On the far right end of the Workspace toolbar, click the Show/Hide Assistants button.



Show/Hide Assistants

Note: This button acts as a toggle such that, when you press it again, the assistant panes that you just hid reappear in the workspace.

➤ Choose *Window – Assistants – Hide All*.

In either case, all assistant panes disappear from your workspace.

Note: If you choose *Window – Assistants – Show All*, all available assistant panes appear in your session window. You can see the complete list of available assistant panes on the submenu that appears when you choose *Window – Assistants*.

Toggling Assistant Panes

To toggle assistant panes off, do the following:

➤ On your keyboard, press the `F11` key.

Any assistant panes you had open in your workspace vanish from the session window.

When you press `F11` again, the assistant panes you had open in your workspace appear once again in the session window.

Note: `Ctrl+F11` can also be used to toggle all toolbar display, and `Shift+F11` will toggle both toolbar and assistant display.

Modifying the Size or Location of an Assistant Pane

How you modify the size or change the location of an assistant pane depends on whether the pane is docked or floating:

- Modifying the Size of a Docked Assistant Pane on page 241
- Modifying the Size of a Floating Assistant Pane on page 242
- Changing the Location of a Docked Assistant Pane on page 242
- Changing the Location of a Floating Assistant Pane on page 242

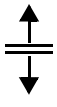
Modifying the Size of a Docked Assistant Pane

To modify the size of a docked assistant pane, do the following:

➤ Click-drag any edge.

Tip

You can identify an edge that you can move when the mouse cursor appears as two parallel lines with outward-facing arrows:

 at vertical edges  at horizontal edges

Your assistant pane appears taller or shorter, wider or narrower, according to how you reposition the edge.

Modifying the Size of a Floating Assistant Pane

To modify the size of a floating assistant pane, do the following:

➤ Click-drag any edge or corner.

Your assistant pane appears taller or shorter, wider or narrower, according to how you reposition the edge or corner.

Changing the Location of a Docked Assistant Pane

To change the location of a docked assistant pane, do the following:

➤ Drag-and-drop the pane by its title bar to a new location within the session window.

The program adjusts the size and location of other items in the window to fit the assistant pane in its new location.

Important

If the new location falls outside the session window, the assistant pane becomes floating (see Changing the Location of a Floating Assistant Pane on page 242.). You can drag-and-drop a floating pane back into your session window to dock it. See also "Docking or Floating an Assistant Pane" on page 237 for additional information.

Changing the Location of a Floating Assistant Pane

To change the location of a floating assistant pane, do the following:

➤ Drag-and-drop the pane by its title bar to a new location.

Tip

If you drag the floating assistant pane to a location within your session window, the pane will become docked by default. You can prevent a floating pane from docking by holding down the *Ctrl* key during the drag-and-drop operation.

If your pane becomes docked when you change its location, the program adjusts the size and location of other items in the window to fit the assistant pane in its new location.

Important

If the new location falls outside the session window, the assistant pane becomes floating. You can drag-and-drop a floating pane back into your session window to dock it.

See "Docking or Floating an Assistant Pane" on page 237 for additional information.

Modifying the Length or Location of a Toolbar

To modify the length of a horizontally-oriented toolbar, do the following:

➤ Click-drag its handle to the left or right.

Tip

You can identify the handle of a toolbar when the mouse cursor appears as four outward-facing arrows:

The toolbar appears longer or shorter depending on whether you dragged its handle to the left or to the right. The program adjusts the length of any neighboring toolbars so that they fit the length of the session window.

To modify the height of a vertically-oriented toolbar, do the following:

➤ Click-drag its handle up or down.

The toolbar appears taller or shorter depending on whether you dragged its handle up or down. The program adjusts the length of any neighboring toolbars so that they fit the height of the session window.

To change the location of a toolbar, do the following:

➤ Drag-and-drop the toolbar by its handle to a new location along any edge of the session window (top, left, right, bottom).

The program makes room at the new location for the relocated toolbar.

The modifications you make last for the duration of the current session unless you save the modified layout as a custom workspace (see "Saving a Workspace" on page 244).

Saving a Workspace

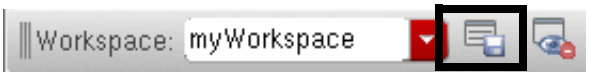
You can save a custom workspace that is based on an existing, default workspace or a new version of a default workspace. For example, you could create a *Constraints_Schematics* workspace to complement the existing *Constraints* workspace or you could save a new version of the *Constraints* workspace.

Caution

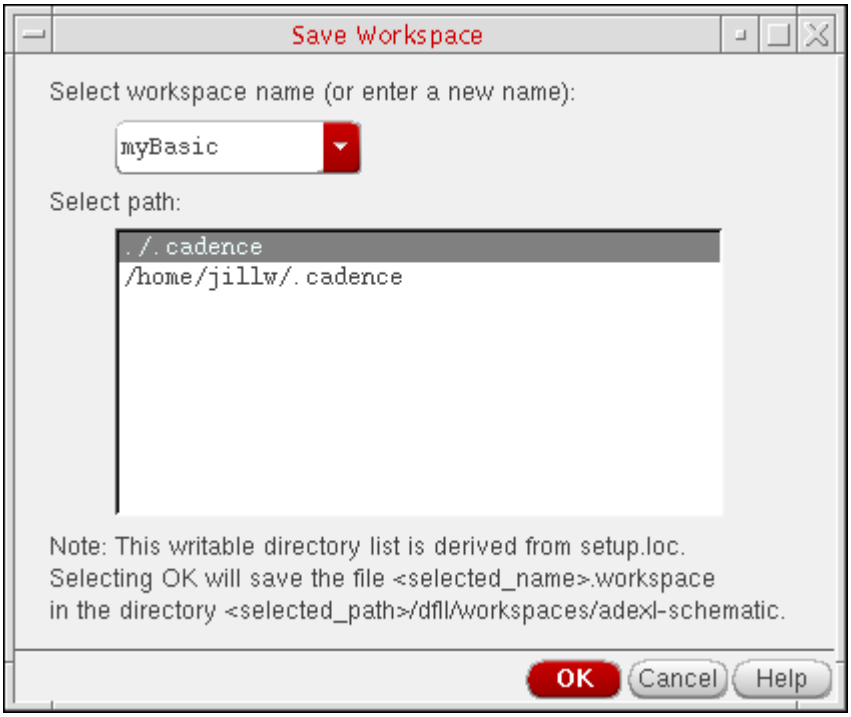
If you save a new version of a default workspace, you might not be able to retrieve the original.

To save your workspace, do the following:

1. On the Workspace toolbar, click the Save Workspace icon.

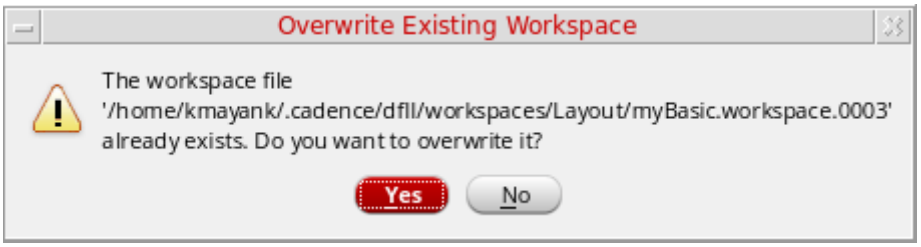


The Save Workspace form appears.



2. In the *Select workspace name* drop-down combo box, either select a workspace name or type a new name.
3. Select the *Overwrite existing workspace* check box if you want to overwrite the existing workspace.

   If the check box is not selected and the name of the workspace is same as that of an existing workspace, the following message is displayed.



Click *Yes* to continue.

Note: You can overwrite the existing workspace only if you have the required permissions. In case you do not have the required permissions, an error message is displayed.

4. In the *Select path* list area, select a location.
5. Click *OK*.

   A message appears in the output area of your Command Interpreter Window indicating the name and location where the program saved your workspace. For example:

```
Saved Workspace "myBasic" to
"./.cadence/dfII/workspaces/adexl-schematic/myBasic.workspace"
Saved Workspace "myBasic" to
"/home/user/.cadence/dfII/workspaces/adexl-schematic/myBasic.workspace
```

See also

- Workspace Search Order on page 222
- Loading a Custom Workspace on page 246

Loading a Custom Workspace

You can load a custom workspace using the Workspaces toolbar or the *Window* menu in your session window.

To load a custom workspace using the Workspace toolbar, do the following:

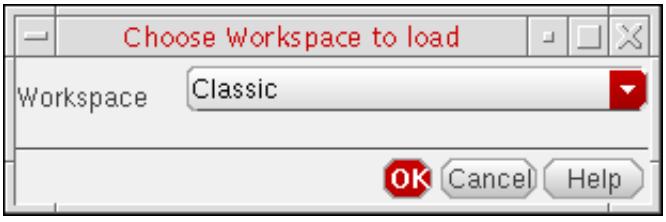
➤ In the *Workspace* drop-down combo box, select a workspace.

The program configures your session window using the workspace you selected.

To load a custom workspace using the *Window* menu, do the following:

1. Choose *Window – Workspaces – Load*.

   The Choose Workspace to Load form appears.



2. In the *Workspace* drop-down combo box, select a workspace.
3. Click *OK*.

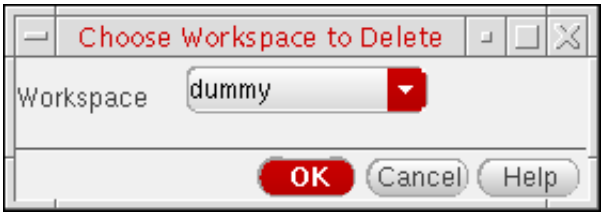
   The program configures your session window using the workspace you selected.

Deleting a Custom Workspace

To delete a custom workspace, do the following:

1. Choose *Window – Workspaces – Delete*.

   The Choose Workspace to Delete form appears.



2. In the *Workspace* drop-down combo box, select the workspace you want to delete.
3. Click *OK*.

   The program deletes the workspace. If you delete the current workspace, the program displays the default workspace.

Customizing a Cadence Workspace

Important

You must have the necessary read-write permissions in the area where you want to save a custom workspace. Typically, you do not have write permission to the Cadence installation area but you do have write permission to your local directory tree. You might also have write permission to the area where your site-specific customizations are stored.

To customize a Cadence workspace and save it, do the following:

1. Select a Cadence workspace.
2. Modify the Cadence workspace to your specifications.
3. Save your custom workspace.

Tip

If you want all users to have access to your customized workspace, the directory to which you save it must be in the site-specific area at your location.

You can check your `setup.loc` file to verify that the search settings are correct for displaying your customized workspace (see "Workspace Search Order" on page 222). For example, if you want a customized workspace to appear for all users, and the name of the customized workspace is the same as a Cadence workspace, you must edit the `setup.loc` file so that it checks your site-specific area first. If your `setup.loc` file checks the Cadence installation area first, the program will find the Cadence workspace first.

If you save a customized workspace to your site-specific area using a name that matches a Cadence workspace, and you want the customized configuration to appear for all users, you must edit the `setup.loc` file such that it checks your site-specific area first. If your `setup.loc` file checks the Cadence installation area first, the program will find the Cadence workspace first.

Reverting to a Saved Workspace

To revert to the last version of the current workspace saved to disk, do the following:

➤ Choose *Window – Workspaces – Revert to Saved*.

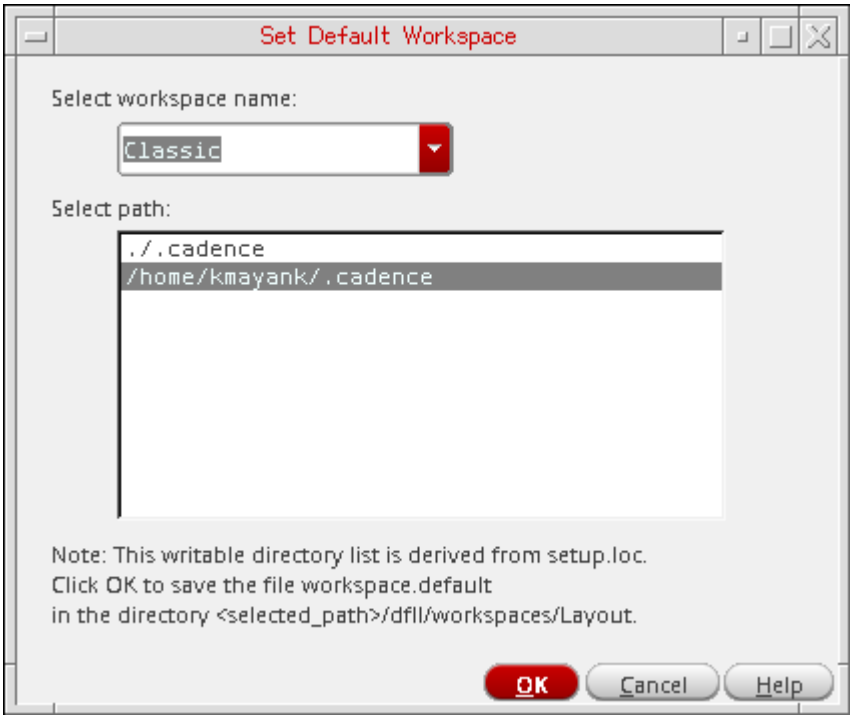
The last-saved version of the current workspace appears in the session window.

Setting the Default Workspace for an Application

To select a current workspace as the default for a particular application, do the following:

1. Choose *Window – Workspaces – Set Default*.

   The Set Default Workspace form is displayed.



2. From the *Select workspace name* drop-down combo box, select the workspace you want to use as the new default.

   This workspace will appear for each subsequent invocation of the current application or view type.

3. Optionally, select the path where you want to save the default workspace specifications.

   All writable locations in your CSF (Cadence Search File) will be listed.

   Note: Assuming that your home directory has been set up as a member of the CSF, the workspace will be saved to `$HOME` by default. However, you may want to change this to `./.cadence`, or another writable CSF location, so that the default is only applied to the current design.

4. Click the *OK* button to set the new default workspace for the current application.

11

Using Bookmarks and Views

You can bookmark design views and return to them during the current or future sessions. You can create a bookmark of a single view or a composite bookmark of all tabs in your session window. You can restore bookmark design views and import/export bookmark files. You can designate whether your bookmarks appear only on the bookmarks menu or also on the *Bookmarks* toolbar. You can select bookmarks from the menu or from the Bookmarks toolbar.

You can use bookmarks with any of the following Cadence applications:

- Virtuoso Schematic Editor L/XL
- Virtuoso Layout Suite Viewer/XL/EXL
- Virtuoso ADE Explorer
- Virtuoso ADE Assembler
- Virtuoso ADE Verifier
- Virtuoso Text Editor

Note: You can also access bookmark functionality from the CIW (*File – Bookmarks*). When opening a bookmark from the CIW, the design will be opened in a new session window rather than opening it as a new tab in the current session window.

The program stores bookmark information in `./.cadence/dfII/bookmarks/`*`username`*`.bookmarks`. The program searches for global bookmark information in `global.bookmarks`. For more information on storage, see File Storage.

Note: If the program encounters any files with the `.bkm` extension (such as the old format global `bookmarks.bkm` file), it converts them to the `.bookmarks` format.

You can read more about bookmarks in the following sections:

Adding the Bookmarks Toolbar to a Session Window

To add the Bookmarks toolbar to your current session window, do one of the following:

- Right-click in the toolbar area of the session window and click to mark the *Bookmarks* item on the pop-up menu.
- Choose *Window – Toolbars – Bookmarks*.

  The program adds the Bookmarks toolbar to your current session window.

  Note: Other session windows are not affected by this addition.

See also "Adding Assistant Panes and Toolbars to a Workspace" on page 232.

Bookmarking Designs

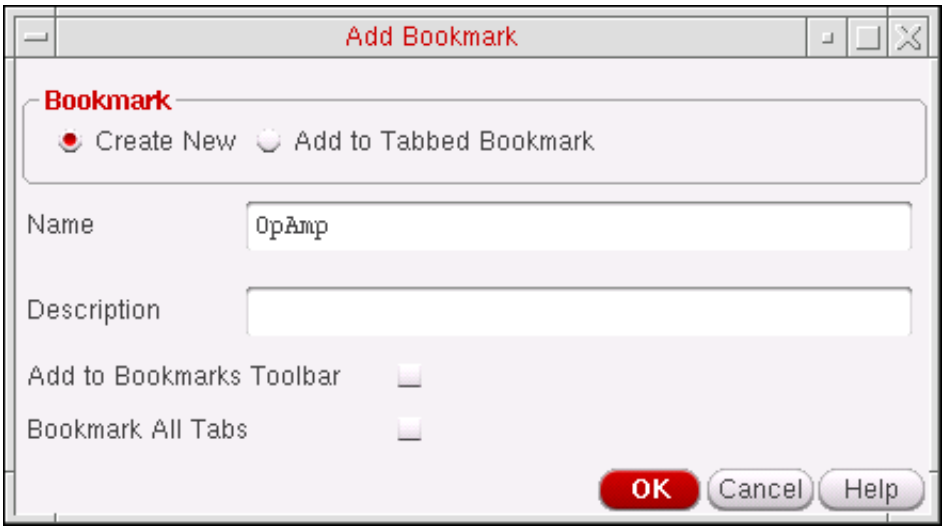
You can bookmark any number of cellviews.

Note: See also Adding Bookmarks to Existing Composite Bookmarks for information on how to add further bookmarks to existing composites (collections of bookmarks).

To add a bookmark for the current design or a collection of open designs (creating a composite bookmark), do the following:

1. In the session window, choose *File – Bookmarks – Add Bookmark*.

   The Add Bookmark form appears.



   The name of the current design cellview appears in the *Name* field.

   Note: You can also right-click over any existing session window tab, and select *Add Bookmark* to display the Add Bookmark form.

2. Select the *Create New* bookmark option.

   Note: For information on using the *Add to Tabbed Bookmark* option see Adding Bookmarks to Existing Composite Bookmarks.

3. (Optional) In the *Name* field, type a different name for your bookmark.

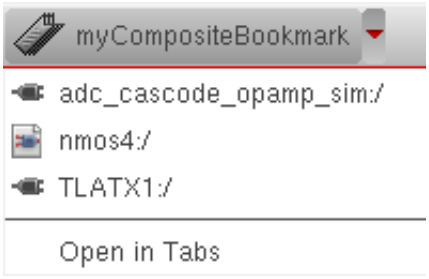
4. (Optional) In the *Description* field, type a description for your bookmark.

   This text appears in the *Description* column of the Bookmarks Manager window.

5. (Optional) If you want your bookmark to also appear on the Bookmarks toolbar, check the *Add to Bookmarks Toolbar* check box.

   Note: There is no limit to the number of bookmarks you can add to the Bookmarks toolbar. However, when the Bookmarks toolbar area is full, a ">>" pull-down appears on the right side of the toolbar so that you can select additional bookmarks that do not appear because of space restrictions.

6. (Optional) If you want to add a composite bookmark for all the tabs in your session window, check the *Bookmark All Tabs* check box.



   A composite bookmark lists all the views that it comprises of. You can choose to open one bookmark or all of them (*Open in Tabs*).

7. Click *OK*

   The bookmark will be created and optionally added to the *Bookmarks* toolbar. Otherwise, the bookmark will be available from the bottom of the *File - Bookmarks* menu options in both the CIW and the current session window, and also from the Bookmarks Manager form (see Managing Bookmarks on page 263).

See also

- Restoring a Bookmark on page 258.

Restoring a Bookmark

You can restore single or composite bookmarks using the *File – Bookmarks* menu or from the Bookmarks toolbar. A composite bookmark is a bookmark created from all tabs in a session window (see "Bookmarking Designs" on page 256). When you restore a composite bookmark, you can restore the entire set of tabs or just one of the tabs that makes up the set.

The following symbols identify single bookmarks (schematic, symbol, layout views):

The following symbol identifies a composite bookmark:

- Restoring a Single Bookmark from the Bookmarks Toolbar on page 258
- Restoring a Composite Bookmark from the Bookmarks Toolbar on page 259
- Restoring a View from a Composite Bookmark on the Bookmarks Toolbar on page 259
- Restoring a Single Bookmark Using the File Menu on page 260
- Restoring a Composite Bookmark Using the File Menu on page 261
- Restoring a Cellview from a Composite Bookmark Using the File Menu on page 262

Note: You can also access bookmark functionality from the CIW (*File – Bookmarks*). When opening/restoring a bookmark from the CIW, the design will be opened in a new session window rather than opening it as a new tab in the current session window.

Restoring a Single Bookmark from the Bookmarks Toolbar

To restore a single bookmark from the Bookmarks toolbar, do the following:

➤ Click the bookmark name.

The bookmarked view appears in your session window.

Note: If you do not see your single bookmark on the Bookmarks toolbar, you might find it on the *File – Bookmarks* menu instead. See "Restoring a Single Bookmark Using the File Menu". See also "Managing Bookmarks" on page 263.

Restoring a Composite Bookmark from the Bookmarks Toolbar

When you restore a composite bookmark, you can choose to restore the entire set of views that make up the composite bookmark, or just one view from the composite (see Restoring a View from a Composite Bookmark on the Bookmarks Toolbar for the latter).

To restore all views that comprise a composite bookmark from the Bookmarks toolbar, do the following:

1. Click the down arrow to the right of the composite bookmark name.

   A drop-down menu appears.

   Note: If you click directly on the composite bookmark name only the first view listed in the composite bookmark will be opened in a new tab.

2. Select *Open in Tabs*.

   Each view in the composite bookmark will open in an individual tab in your session window.

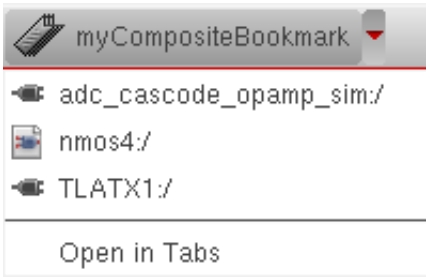
Note: If you do not see your composite bookmark on the Bookmarks toolbar, you might find it on the *File – Bookmarks* menu instead. See "Restoring a Composite Bookmark Using the File Menu". See also "Managing Bookmarks" on page 263.

Restoring a View from a Composite Bookmark on the Bookmarks Toolbar

To restore one view from a composite bookmark on the Bookmarks toolbar, do the following:

1. Click the down arrow to the right of the composite bookmark name.

   A drop-down menu appears.



   Note: If you click directly on the composite bookmark name only the first view listed in the composite bookmark will be opened in a new tab.

2. Select the view you want to restore.

The bookmarked view appears in your session window.

Note: If you do not see your composite bookmark on the Bookmarks toolbar, you might find it on the *File – Bookmarks* menu instead. See "Restoring a Cellview from a Composite Bookmark Using the File Menu". See also "Managing Bookmarks" on page 263.

Restoring a Single Bookmark Using the File Menu

To restore a single bookmark using the *File* menu, do the following:

1. In either the session window or the Command Interpreter Window, choose *File – Bookmarks*.

   A submenu appears.

2. If the bookmark appears on the Bookmarks toolbar, choose *Toolbar Bookmarks*.

   A pull-right menu appears.

3. Select the bookmark you want to restore.

   The bookmarked view appears in your session window.

See also "Managing Bookmarks" on page 263.

Restoring a Composite Bookmark Using the File Menu

To restore a composite bookmark using the *File* menu, do the following:

1. In either the session window or the Command Interpreter Window, choose *File – Bookmarks*.

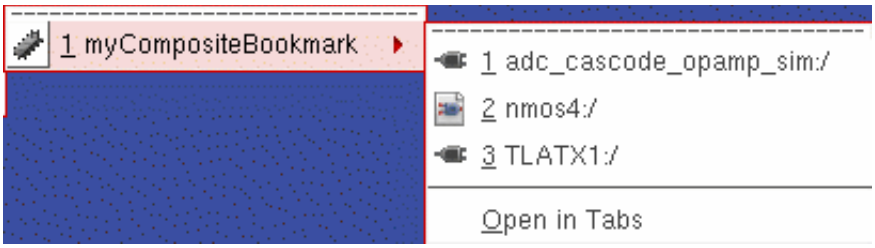
   A submenu appears.

2. If the bookmark appears on the Bookmarks toolbar, choose *Toolbar Bookmarks*.

   A pull-right menu appears.

3. Select the bookmark you want to restore.

   A pull-right menu of views appears.



4. At the bottom of this menu, select *Open in Tabs*.

   The bookmarked views appear on new tabs in your session window.

See also "Managing Bookmarks" on page 263.

Restoring a Cellview from a Composite Bookmark Using the File Menu

To restore a cellview from a composite bookmark using the *File* menu, do the following:

1. In either the session window or the Command Interpreter Window, choose *File – Bookmarks*.

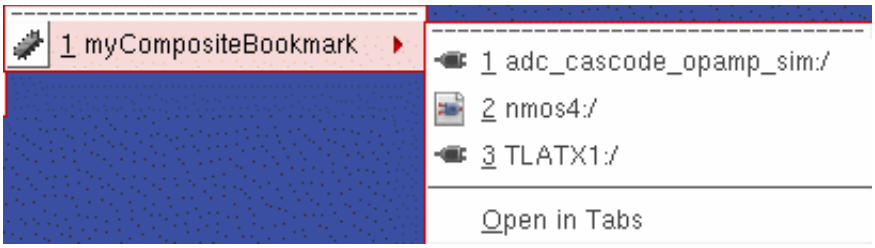
   A submenu appears.

2. If the bookmark appears on the Bookmarks toolbar, choose *Toolbar Bookmarks*.

   A pull-right menu appears.

3. Select the bookmark you want to restore.

   A pull-right menu of cellviews appears.



4. Select the cellview you want to restore.

   The bookmarked cellview appears in your session window.

See also “Managing Bookmarks” on page 263.

Managing Bookmarks

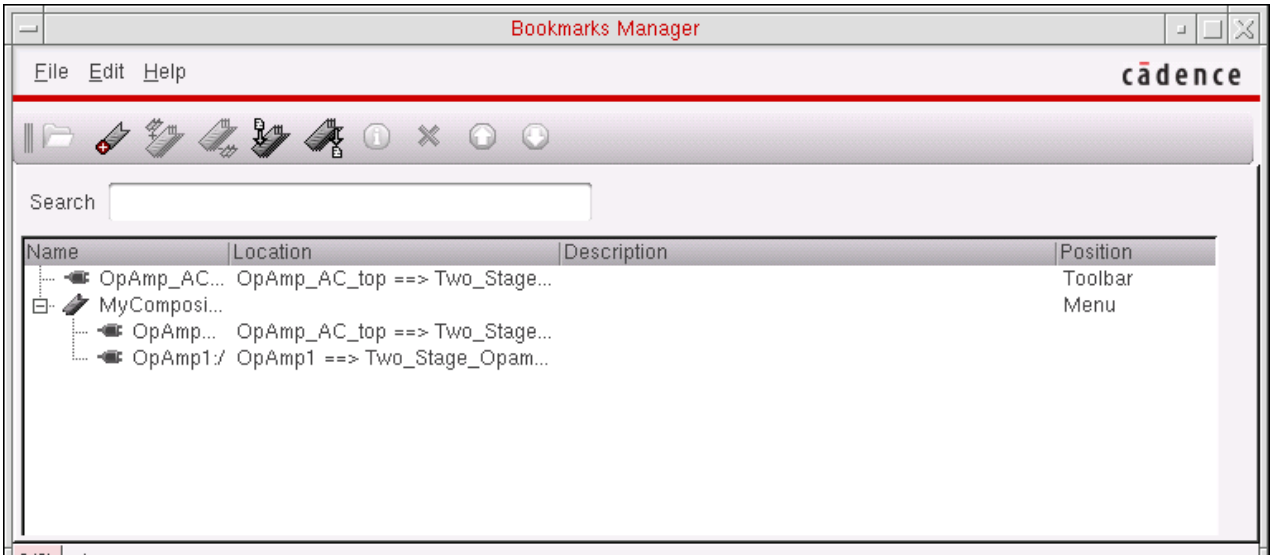
Important

For information on how to specifically manage composite (collections of) bookmarks see Managing Composite Bookmarks on page 275.

To begin managing your bookmarks, do the following:

➤ In your session window or in the Command Interpreter Window (CIW), choose *File – Bookmarks – Manage Bookmarks*.

The Bookmarks Manager window appears.



The Bookmarks Manager window contains the following information about your bookmarks:

Column	Information
Name	Bookmark name
Location	Cell name following by its `lib__cell__view` designation
Description	Bookmark's description
Position	*Toolbar* if the bookmark appears on the Bookmarks toolbar; otherwise, *Menu*

You can perform the following tasks related to managing your bookmarks:

- Adding a Bookmark on page 265
- Importing Bookmarks on page 266
- Exporting Bookmarks on page 268
- Deleting Bookmarks on page 270
- Editing Bookmark Properties on page 271
- Opening Bookmarked Views on page 272
- Searching for Bookmarks on page 273
- Closing the Bookmarks Manager Window on page 273

See also

- Viewing Help for the Bookmarks Manager on page 273
- Viewing Cadence Documentation from the Bookmarks Manager on page 274

Adding a Bookmark

To add a bookmark, do the following:

➤ In the Bookmarks Manager window, choose *Edit – Add*.

Tip

Alternatively, you can click the Add Bookmark icon on the toolbar:

The Add Bookmark form appears so that you can add a bookmark to the current tab in the current session window (see "Bookmarking Designs" on page 256).

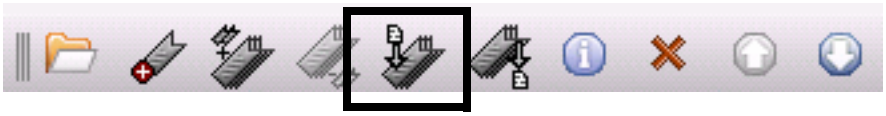
Importing Bookmarks

To import a set of bookmarks from a valid bookmark file, do the following:

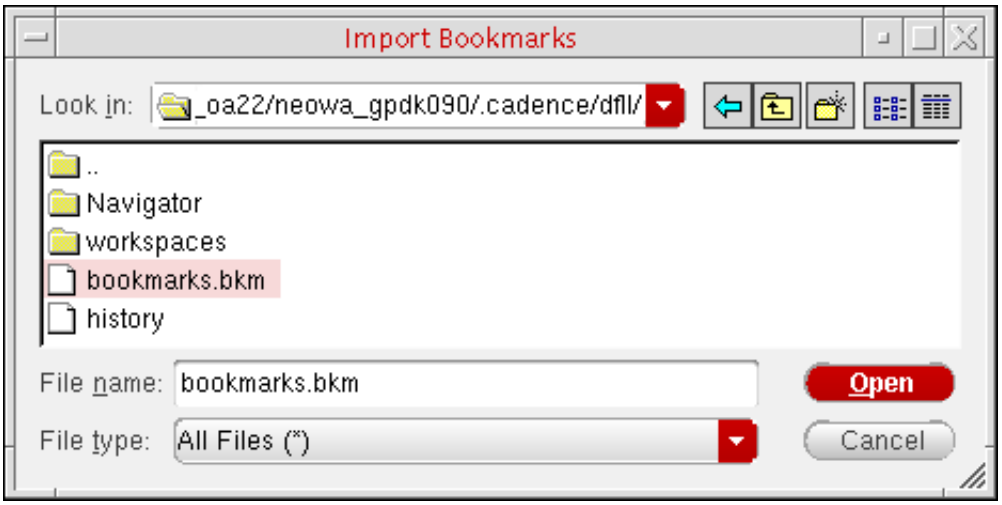
1. In the Bookmarks Manager window, choose *File – Import*.

Tip

Alternatively, you can click the Import Bookmarks icon on the toolbar:



The Import Bookmarks form appears.



2. Navigate to and select a valid bookmark file.
3. Click *Open*.

   The program imports bookmarks from the selected file.

   Note: The program does not filter out duplicate bookmarks.

Setting Import Bookmark Behavior

You can choose to ignore any duplicate bookmarks that are found in an imported bookmark list.

To control imported bookmark behavior use the boolean environment variable `importDuplicateBookmarks`. This variable has a default setting of `t`, for example:

```
envGetVal ("ddserv" "importDuplicateBookmarks") = t
```

When set to `t`, all bookmarks that are found to be duplicates will be imported. That is, any bookmarks that share the same name and callbacks between the imported bookmark file and the current bookmark file will still be imported.

However, when `importDuplicateBookmarks` is set to `nil`, as shown below, any bookmarks that are found to be duplicate, in the file to be imported, will not be imported into the current Bookmarks Manager bookmark list.

```
envSetVal ("ddserv" "importDuplicateBookmarks" 'boolean nil)
```

Note: A bookmark from an imported bookmark file could be found to be a duplicate with either a bookmark in your local bookmark file or with a bookmark that is included in the global bookmarks file.

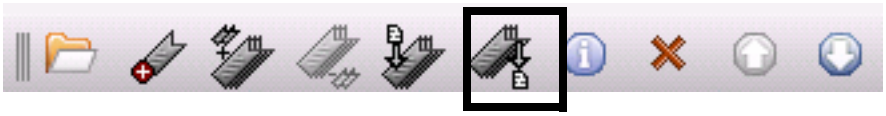
Exporting Bookmarks

To export your bookmarks to a file, do the following:

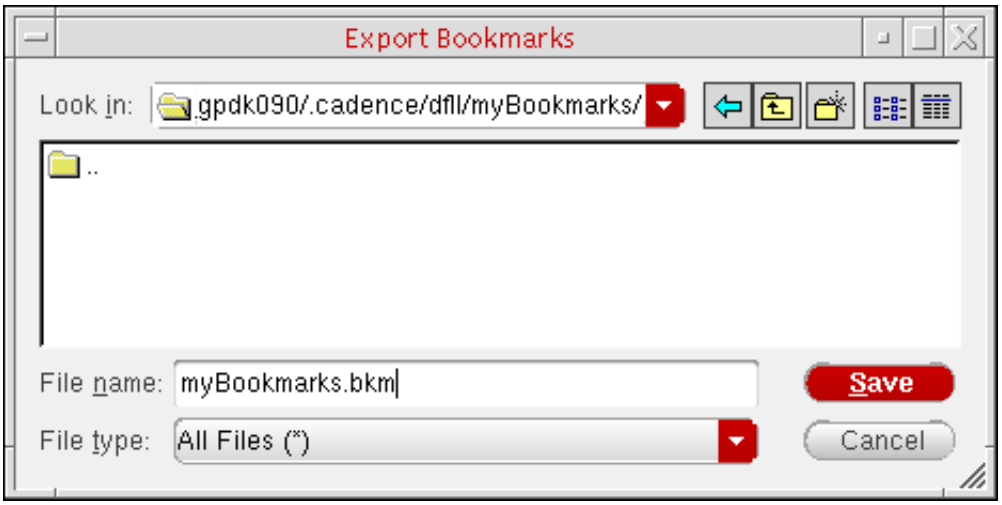
1. In the Bookmarks Manager window, choose *File – Export*.

Tip

Alternatively, you can click the Export Bookmarks icon on the toolbar:



The Export Bookmarks form appears.



2. Navigate to the location where you want to save your bookmarks file.
3. In the *File name* field, type a name for your bookmarks file.
4. Click *Save*.

Reordering Bookmarks

To reorder your bookmarks, do the following:

1. In the Bookmark Managers window, select the bookmark entry that you want to move up or down the bookmark list.
2. On the Bookmark's toolbar, select either the *Move up...* or *Move down...* icon to re-position the selected bookmark.

Note: If you want to change the bookmark display order of composite bookmarks, you will only be able to re-position them within the composite (collective) bookmark.

Deleting Bookmarks

To delete one or more bookmarks, do the following:

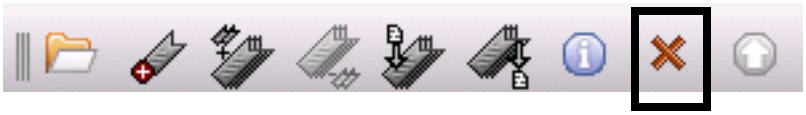
1. In the Bookmarks Manager window, select one or more bookmark names.

   You can select a single bookmark, a composite bookmark, a single view that is part of a composite bookmark, or any combination of these.

2. Choose *Edit – Delete*.

Tip

Alternatively, you can click the Delete Bookmarks icon on the toolbar:



The program deletes the bookmarks you selected.

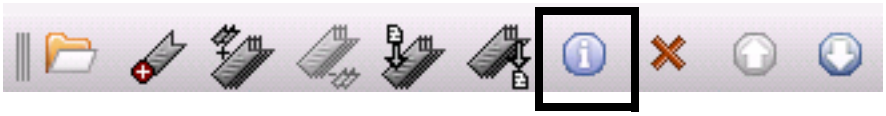
Editing Bookmark Properties

To change the name, description, or position of a bookmark, do the following:

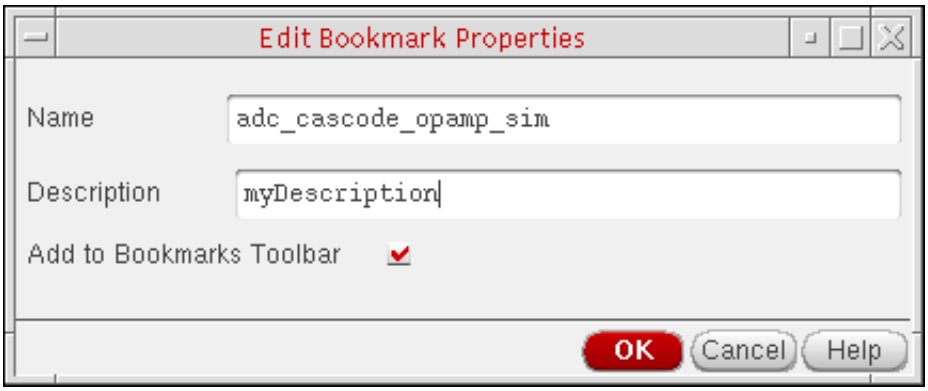
1. In the Bookmarks Manager window, choose *Edit – Properties*.

Tip

Alternatively, you can click the Edit Bookmark Properties icon on the toolbar:



The Edit Bookmark Properties form appears.



2. (Optional) In the *Name* field, edit the name of your bookmark.
3. (Optional) In the *Description* field, edit the description of your bookmark.
4. If you want your bookmark to appear on the Bookmarks toolbar, make sure there is a mark in the *Add to Bookmarks Toolbar* check box.

   If you do not mark the *Add to Bookmarks Toolbar* check box, your bookmark appears on the *File – Bookmarks* submenus only. (You can choose *File – Bookmarks* in the session window or in the Command Interpreter Window.)
5. Click *OK*.

   The changes you made appear in the Bookmarks Manager window.

Opening Bookmarked Views

To open a bookmarked view or views, do the following:

1. In the Bookmarks Manager window, select one or more bookmarks.
2. Choose *File – Open*.

Tip

Alternatively, you can click the Open Bookmarks icon on the toolbar:

The program opens the items you selected. Each selected item appears in a new session window.

Note: If you expand a composite bookmark before making your selections, be aware that each selected line in the Bookmarks Manager window counts as a selected item (up and down movement is restricted within the composite bookmark itself). The program opens each selected item in its own session window. For example, if you expand a composite bookmark containing two cellviews and select all three lines in the Bookmarks Manager window like this:

when you choose *File – Open*, three session windows appear:

- One containing the composite bookmark: *overview follower schematic* on one tab and *overview diffAmp schematic* on another tab
- One containing the *overview follower schematic*
- One containing the *overview diffAmp schematic*

Searching for Bookmarks

To search for a bookmark, do the following:

➤ In the *Search* field, type the search string.

The program looks for matching strings in the *Name*, *Location*, and *Description* columns.

As you type, the Bookmarks Manager filters all bookmarks such that only those whose names, locations, or descriptions contain the search string appear in the window. If the program finds a matching string in a bookmark that is part of a composite bookmark, the program displays the expanded tree for that composite bookmark so that you can see the matching string.

You can restore the complete list by deleting the search string.

You can perform the following tasks on the selected bookmark or bookmarks:

- Opening Bookmarked Views on page 272
- Editing Bookmark Properties on page 271
- Deleting Bookmarks on page 270

Closing the Bookmarks Manager Window

To close the Bookmarks Manager window, do the following:

➤ In the Bookmarks Manager window, choose *File – Close*.

Tip

Alternatively, you can close the window by pressing *Alt+F4*.

Viewing Help for the Bookmarks Manager

To view Help for the Bookmarks Manager, do the following:

➤ In the Bookmarks Manager window, choose *Help – Current Context*.

Viewing Cadence Documentation from the Bookmarks Manager

To view general Cadence documentation from the Bookmarks Manager, do the following:

➤ In the Bookmarks Manager window, choose *Help – Cadence Documentation*.

Managing Composite Bookmarks

You can manage existing composite (tabbed) bookmarks with the following bookmark features:

- Adding Bookmarks to Existing Composite Bookmarks on page 275
- Creating Composite Bookmarks from Existing Bookmarks on page 276
- Detaching Bookmarks from Existing Composite Bookmarks on page 280

Adding Bookmarks to Existing Composite Bookmarks

To add another bookmark to an existing collection of bookmarks:

1. In the session window, choose *File – Bookmarks – Add Bookmark*.

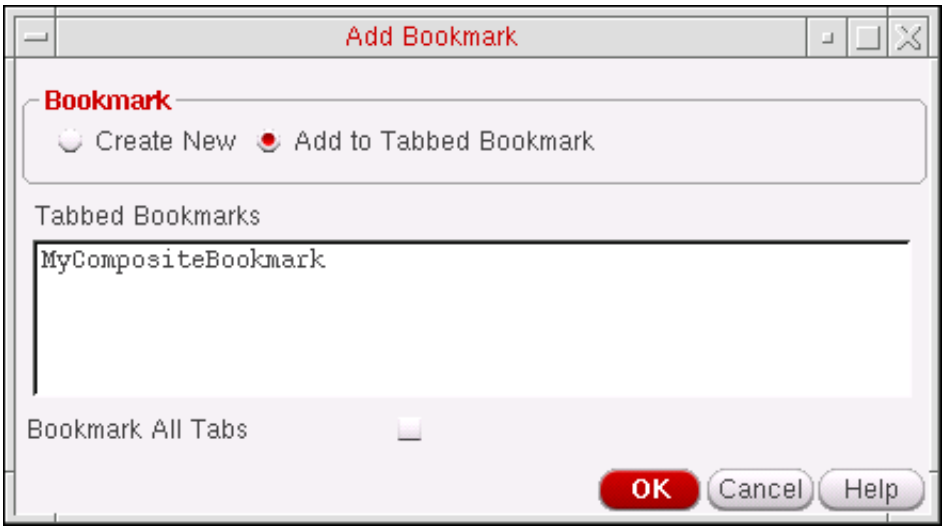
   The Add Bookmark form appears.

   The name of the current design cellview appears in the *Name* field.

   Note: You can also right-click over any existing session window tab, and select *Add Bookmark* to display the Add Bookmark form.

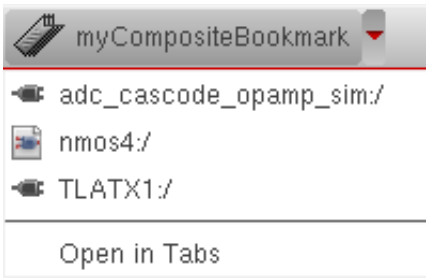
2. Select the *Add to Tabbed Bookmark* option.

   The Add Bookmark form updates to reflect the new setting.



   All existing tabbed bookmarks are listed in the *Tabbed Bookmarks* section.

3. Select the tabbed bookmark that you want to add the current bookmark to.
4. (Optional) If you want to add all the current tabs in your session window to the selected tabbed bookmark, check the *Bookmark All Tabs* check box.



5. Click *OK.*

   The bookmark will be added to the selected composite bookmark and displayable in the *Bookmarks Manager* when the composite is expanded.

Creating Composite Bookmarks from Existing Bookmarks

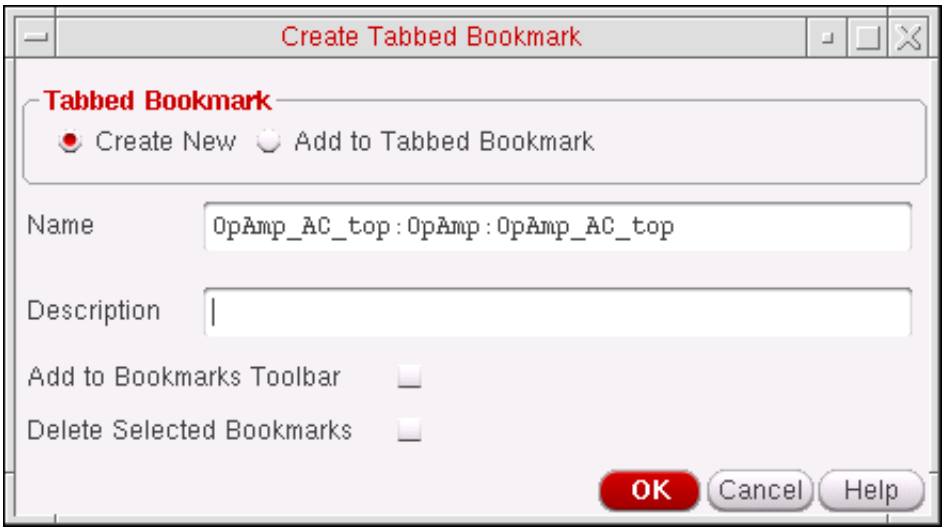
You can create composite (tabbed) bookmarks from one or more existing bookmarks. These bookmarks can be either non-composite bookmarks and/or bookmarks that are already part of existing composites.

To create a composite bookmark from existing bookmarks:

1. In the *Bookmarks Manager*, select the bookmarks that you want to create a composite bookmark for (or to add to an existing composite bookmark).
2. Select *Edit – Create Tabbed Bookmark*.

   Note: You can also use the *Create Tabbed Bookmark* option on the *Bookmarks Manager* toolbar.

The Create Tabbed Bookmark form is displayed with the selected bookmark names displayed in the *Name* field.



3. Select to either create a new (*Create New*) tabbed bookmark, to contain the selected bookmarks, or an existing tabbed (composite) bookmark (*Add to Tabbed Bookmark*).

- If you selected *Create New* go to step 4.
- If you selected *Add to Tabbed Bookmark* go to step 7.

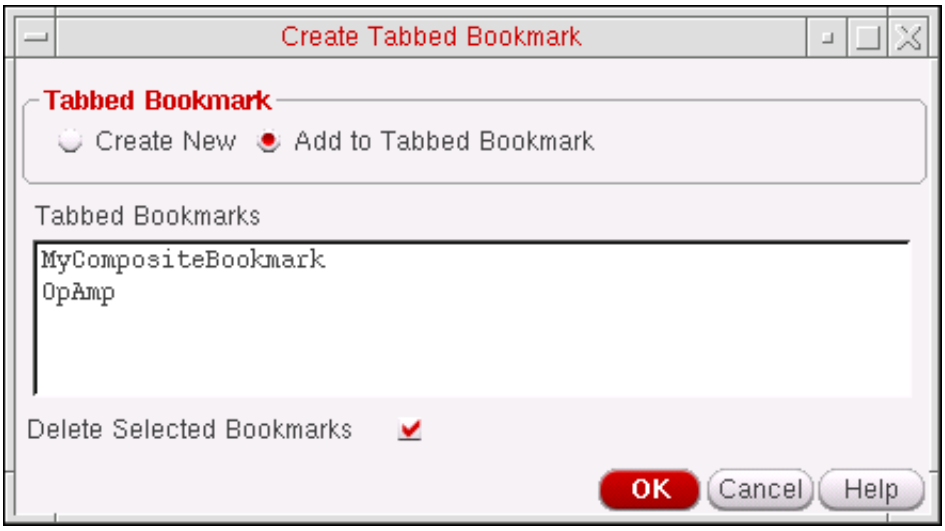
4. (Optional) In the *Name* field, type a different name for your bookmark.
5. (Optional) In the *Description* field, type a description for your bookmark.

   This text appears in the *Description* column of the Bookmarks Manager window.
6. (Optional) If you want your composite bookmark to also appear on the Bookmarks toolbar, check the *Add to Bookmarks Toolbar* check box.

   Note: There is no limit to the number of bookmarks you can add to the *Bookmarks* toolbar. However, when the *Bookmarks* toolbar area is full, a ">>" pull-down appears on the right side of the toolbar so that you can select additional bookmarks that do not appear because of space restrictions.

   Go to step 8.
7. Selecting the *Add to Tabbed Bookmark* option updates the Create Tabbed Bookmark form to display a section that contains all the existing *Tabbed Bookmarks*.

   In this section, choose the existing tabbed bookmark that you want to add the additional, selected, bookmarks to.



8. (Optional) If you also want to remove the bookmarks from their original bookmark location (whether that was originally a single bookmark or a bookmark that was part of a composite), check the *Delete Selected Bookmarks* option.
9. Click *OK*

A new (tabbed) composite bookmark will be created, or the selected bookmarks will be added to an existing tabbed bookmark.

Detaching Bookmarks from Existing Composite Bookmarks

To create simple (single) bookmarks by either removing them or “copying” them from existing tabbed (composite) bookmarks:

1. In the *Bookmarks Manager*, select the bookmarks that you want to detach from an existing composite bookmark.

   Note: By detaching a bookmark from a composite bookmark you are not automatically deleting the bookmark from its composite host (see step 4 below).

2. Select *Edit – Detach Selected Bookmarks* (or select *Detach Selected Bookmarks* from the *Bookmarks Manager* toolbar).

   The Create Simple Bookmarks form is displayed.

3. (Optionally) Check the *Add to Bookmarks Toolbar* option so that the bookmarks to be detached are available from the *Bookmarks* toolbar.
4. (Optionally) Check the *Delete Selected Bookmarks* option if you want to delete the selected bookmarks from their current composite bookmark host.
5. Click the *OK* button to create the simple bookmarks.

   If chosen, the bookmarks will also appear in the *Bookmark* toolbar and/or be deleted from the composite it was initially found in.

Working with Views

See the following sections for more information:

- Saving a View on page 281
- Restoring a Saved View on page 282
- Restoring the Previous View on page 282
- Restoring the Next View on page 282

Saving a View

To save a view, do the following:

1. Choose *View – Save/Restore – Save View*.

   The Save View form appears.

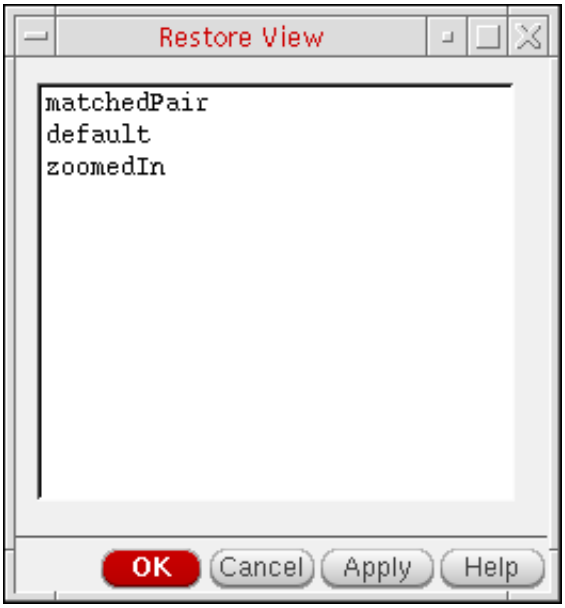
2. In the Name field, type a name for your view.
3. Click *OK*.

Restoring a Saved View

To restore a saved view, do the following:

1. Choose *View – Save/Restore*.

   The Restore View form appears.



2. Select the view you want to restore.
3. Click *OK*.

   The program restores the selected view.

Restoring the Previous View

To restore the previous view, do the following:

➤ Choose *View – Save/Restore – Previous View*.

The program returns to the previous view.

Restoring the Next View

To restore the next view, do the following:

➤ Choose *View – Save/Restore – Next View*.

The program displays the next view.

12

Navigating Cellviews and Hierarchies

You can use the Go toolbar to do the following:

- Sequentially or non-sequentially navigate through a cellview hierarchy.
- Navigate between cells and views in various designs.

Use of Go toolbar cell/cellview navigation will only apply to the current session. If required, you can however choose to bookmark design views from different sessions.

Note: The program will store design navigation information in `<install_directory>.cadence/dfII/history/<username>.history`. For more information on storage see File Storage on page 29.

Important

You should take special care when working with data managed designs. Using the Go toolbar (and bookmarks) to revisit design views could prompt the display of *check out* or *check in* forms, depending on set preferences, prompting for user action. The result of this could be the generation of multiple cellview versions, or incorrectly leaving a cellview in a checked-out state.

You can access the Go toolbar from any of the following Cadence applications:

- Virtuoso Schematic Editor XL
- Virtuoso Layout Suite Viewer/XL/EXL
- Virtuoso ADE Explorer
- Virtuoso ADE Assembler
- Virtuoso ADE Verifier

You can read more about cell and design navigation in the following sections:

- Accessing the Go Toolbar on page 287
- Moving Back through a Design on page 288

Accessing the Go Toolbar

To access the Go toolbar, either:

- Select *Window – Toolbars – Go*.
- Right-click the toolbar area and select *Go* from the pop-up menu.

The icons on the Go toolbar are as follows (in order from left to right):

Icon	Brief Description
Back	■ Click the *Back* icon to move one level back to the previously displayed cell/cellview. ■ Click the *Back* drop-down icon (red triangle) to display a list of cell/cellviews that you can choose to go back to. For more information see Moving Back through a Design. **Note:** The equivalent SKILL command is deBack.
Forward	■ Click the *Forward* icon to move one level forward to the cell/cellview that was displayed prior to the use of the last *Back* command. ■ Click the *Forward* drop-down icon (red triangle) to display a list of cell/cellviews that you can choose to go forward to. For more information see Moving Forward through a Design. **Note:** The equivalent SKILL command is deForward.
Up	The *Up* icon is only activated if you have descended into a hierarchy. Selection of the *Up* icon will move you up one level in the hierarchy. For more information see Moving Up a Design.
Top	The *Top* icon is only activated if you have descended into a hierarchy. Selection of the *Top* icon will move you up to the cell/cellview that represents the top of the hierarchy. For more information see Moving to the Top of a Design.

Moving Back through a Design

Use the *Back* icon to return to the cell/cellview that was displayed immediately prior to the current view. Continued selection of this option allows you to move back through successive views, dependent upon availability.

Use the *Back* drop-down icon (red triangle) to display a list of previously displayed cells/cellviews.

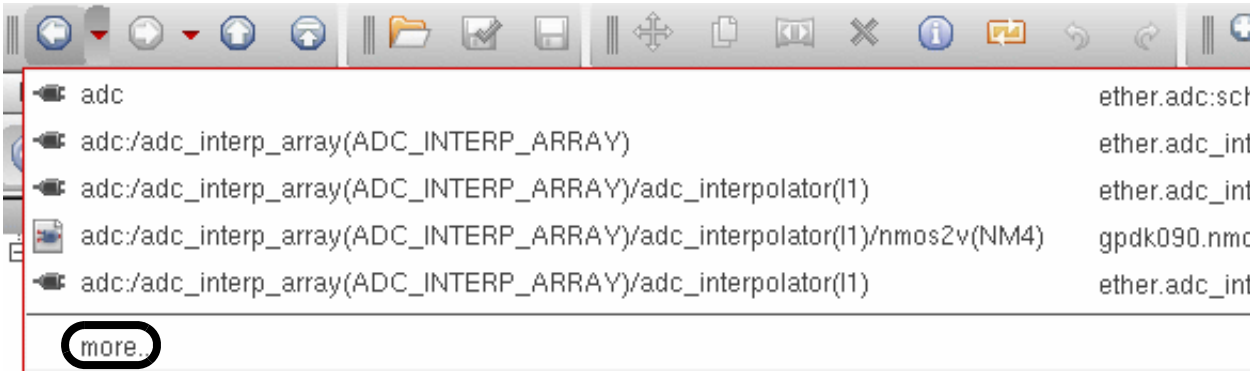
The list presented allows you select a view and navigate straight to it without having to repeatedly scroll back. The cellview at the top of the drop-down list represents the most recently viewed cellview.

The maximum number of cellviews displayed at a time in the drop-down list is five. When the number of possible cellviews to go back to exceeds five, an additional *more* menu item appears on the drop-down list.

To move back through the extended hierarchy, do the following:

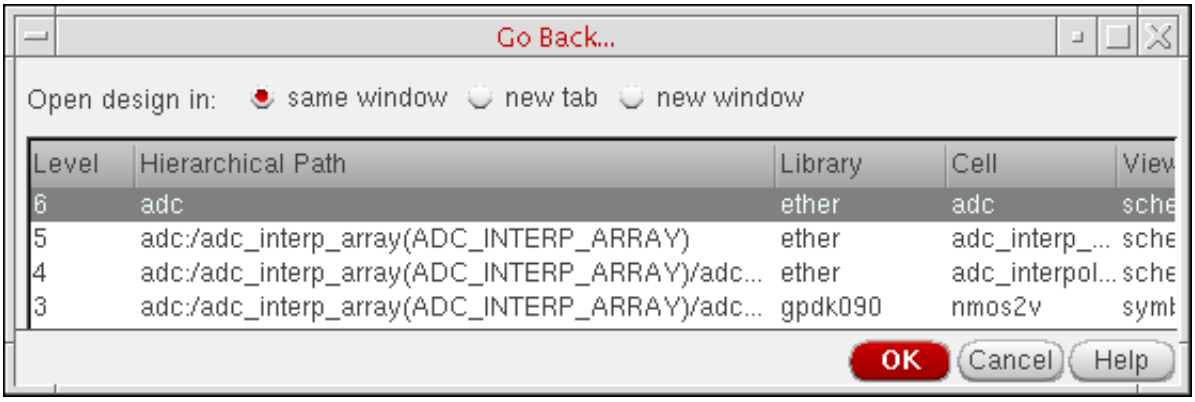
1. On the Go toolbar, click the down-facing red arrow to the right of the Back button.

   A drop-down menu appears.



2. Click *more*.

The Go Back form appears.



Tip

You can resize the form by dragging an edge or corner to the dimensions you want.

3. Select a level.
4. Click *OK*.

   The program restores the selected level on the schematic pane.

Moving Forward through a Design

Use the *Forward* icon to move forward to cells/cellviews that have already been visited using the *Back* icon. Continued selection of this option allows you to move forward through successive views, dependent upon availability.

Note: *Forward* can only be used if *Back* has been used at least once, and if no other cell/cellview has been opened since the last execution of *Back*.

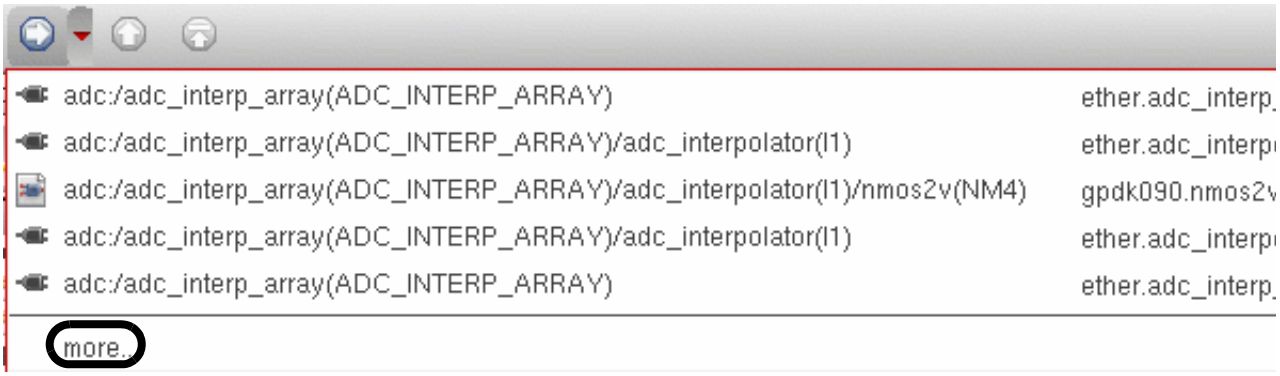
Use the *Forward* drop-down icon (red triangle) to display a list of previously displayed cells/cellviews. The list presented allows you select a view and navigate straight to it without having to repeatedly scroll forward. The cellview at the top of the drop-down list represents the most recently viewed cellview.

The maximum number of cellviews displayed at a time in the drop-down list is five. When the number of possible cellviews to go forward to exceeds five, an additional *more* menu item appears on the drop-down list.

To move forward through the extended hierarchy, do the following:

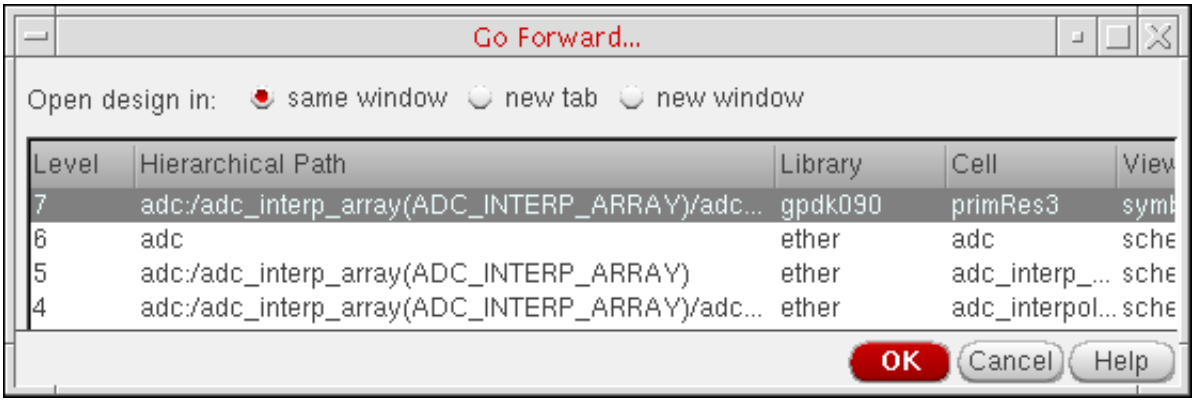
1. On the Go toolbar, click the down-facing red arrow to the right of the Forward button.

   A drop-down menu appears.



2. Click *more*.

   The Go Forward form appears.



Tip

You can resize the form by dragging an edge or corner to the dimensions you want.

3. Select a level.
4. Click *OK*.

   The program restores the selected level on the schematic pane.

Moving Up a Design

Use the *Up* icon to move up a single cell hierarchy and display the cellview that is found one level up. Multiple selection of the *Up* command will take you up the corresponding number of levels in the hierarchy, should they exist.

The *Up* icon is only activated if you have descended into a hierarchy and will be disabled once the top of that hierarchy is reached. Use of *Up* contains you within the current hierarchy.

Note: There is no *Down* icon because you can use the *Back* icon to revisit cellviews.

Moving to the Top of a Design

Use the *Top* icon to open the cellview that represents the top of the hierarchy in the current session window.

The *Top* icon is only activated if you have descended into a hierarchy and will be disabled once the top of that hierarchy is reached.

13

Using Text Windows

Text information appears in text windows. Some examples of text information include design information you request or results of processes you run in read-only mode.

You can perform the following tasks:

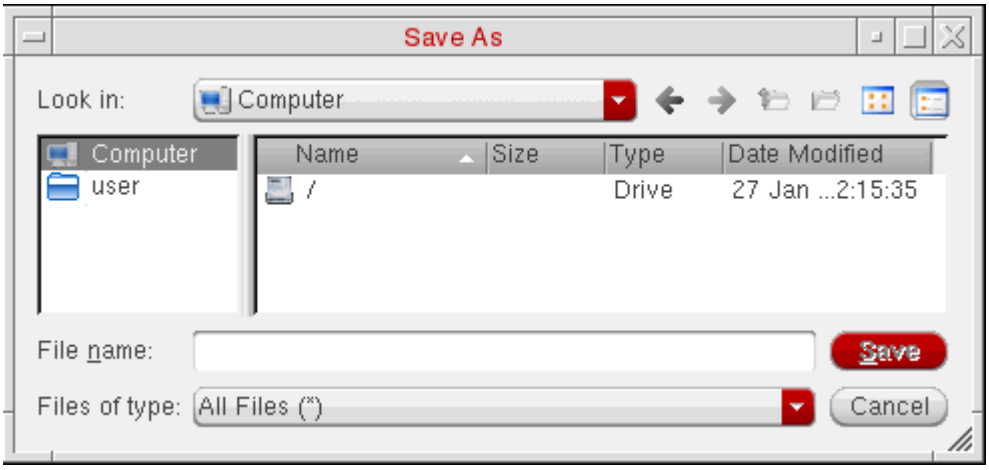
See also Saving Changes if you want to turn off the appearance of the What's New text window every time you start Cadence software.

Saving a Text File

To save the text file, do the following:

1. Choose *File – Save As*.

   The Save As form appears.



2. (Optional) Use the navigation tools (drop-down combo box and toolbar buttons) to specify the directory to which you want to save the text file.

   If you do not specify a directory path, your home directory is used.

3. In the *File name* field, type a name for the text file.
4. Click *Save*.

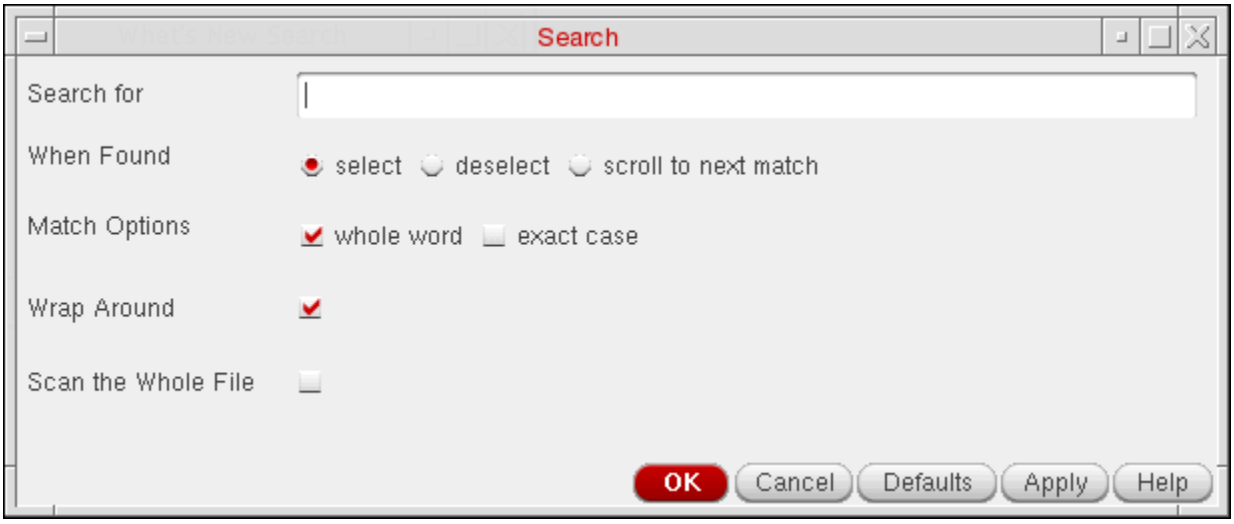
The program writes the specified text file. You can edit the file using a text editor.

Searching the File for Specific Text

To search the file for a specific text string, do the following:

1. Choose *Edit – Find*.

The Search form appears.



2. In the *Search For* field, type the search string.
3. (Optional) Select the *When Found* option.
   - *select* the found text
   - *deselect* the found text
   - *scroll to next match*
4. (Optional) Select one or more of the *Match Options*.
   - *whole word* matches whole words in the text
   - *exact case* matches those strings where the casing is the same as the specified search string
5. (Optional) Select *Wrap Around* to search wrap around text.
6. (Optional) Select *Scan the Whole File* to highlight all occurrence of the search string in the whole file.
7. Click *Apply* to the occurrences of the specified search string.
   - If the specified search string can be found in the text window, the located string is highlighted. Each time you click *Apply*, the next occurrence of the specified string in highlighted until either the end of the text file is reached (if you selected *no* for *Wrap Search*) or you click *OK* (the next step).
   - If the specified search string cannot be found in the text window, or if you selected *no* for *Wrap Search* and you are at the end of the text window and there are no

more occurrences of the specified search string, a *Pattern not found* message appears in the CIW output area.

The Search form remains open.

8. Click *OK* to close the form.

Printing the File

To print the contents of a file in a text window, do the following:

1. Choose *File – Print*.

   The Print Information form appears.

2. In the *Print Command* field, type the UNIX print command plus the name of the file in which you placed (using the *Save As* command) the data.
3. Click *OK*.

The software sends the contents of the named file to the specified printer.

Note: Depending on the application from where the Text Window is opened, the *Print* option can be unavailable.

Refreshing the File

To refresh the contents of the current file, do the following:

➤ Choose *File – Refresh*.
The contents of file are refreshed.

Note: This command differs from the *View – Refresh* option in the *Library Manager* or *File – Refresh* in CIW, which are used to refresh the reading of the `cds.lib` and other technology files.

Viewing Product Version Information

To view product version information, do the following:

➤ Choose *Help – About Virtuoso*.

Product version information appears in a pop-up window.

Note: The *About Virtuoso* option is available on the Text Window that displays What's New content. This window appears when you launch Virtuoso.

Closing a Text Window

To close a text window, do the following:

➤ Choose *File – Close*.

The text window closes.

14

Design Object Limits in OpenAccess

All managed objects in an OpenAccess database are referenced using a 32-bit value, so the theoretical limit of any one type of object in a database is:

```
2^32 = 4,294,967,296
```

However, the actual number of any one type of object is often less than the theoretical limit.

OpenAccess releases before OA22.50 limited the number of string-based objects (`oaShape`s) to this theoretical limit and the total number of named objects to around 2,147,483,648. Starting with OA22.50, these limits on the numbers of objects were increased.

The table below lists the most common design data types and the maximum number of each type that can be safely created.

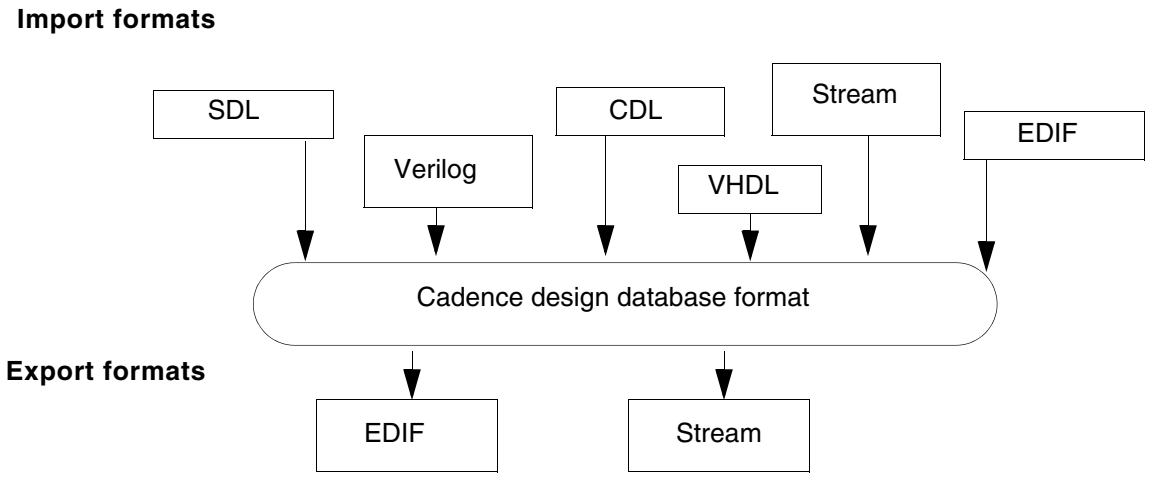
Object Type	DBTypeEnum	Maximum Number of Objects
`oaAppObject`		4,294,967,295
`oaBlockage`	51	4,294,967,295
`oaInst`	18	2,147,483,647 (auto-named and named scalarInsts)
`oaInstTerm`	20	2,147,483,647
`oaNet`	12	2,147,483,647 (auto-named and named scalarInsts)
`oaPin`	28	2,147,483,645
`oaProp`	0	2,147,483,647
`oaRoute`	27	1,431,655,765
`oaShape`	26	4,294,967,295
`oaTerm`	14	2,885,681,151
`oaVia`	64	1,431,655,765

15

Importing and Exporting Designs

The Virtuoso Design Environment is an open system. You can create design data in a number of industry-standard data formats and use Cadence products to work on or complete your design.

The design data translators let you translate a design in one format to a Cadence design database format or take a Cadence design database and translate it into another format. Cadence offers the following translators (your site might or might not offer all of them):



For more information on design data translators, see the following books:

- *Verilog In for Design Framework II User Guide and Reference Manual*
- *Virtuoso EDIF 200 Reader and Writer User Guide*
- *Guide to VHDL In for Design Framework II*
- *Design Data Translator's Reference on OpenAccess*

Importing Designs

To import a design in another format into the Cadence design databse format,

1. From the Command Interpreter Window (CIW), choose *File – Import*.

   A submenu of the available translators on your system appears.

2. Select a translator.

The translator form appears.

Tip

Click *Help* to see more information about that translator menu.

Exporting Designs

To export a design from the Cadence design databse format into another design format,

1. From the CIW, choose *File – Export*.

   A submenu of the available translators on your system appears.

2. Select a translator.

The interface form for the translator appears.

Tip

Click the *Help* button to see more information about that translator menu.

16

Scaling Design Data

You can use the XScale utility to change the scale of your design data. XScale scales objects in Virtuoso Design Environment cellviews according to the magnification factor you specify. You can apply XScale to views or view types within a certain cell or an entire library.

In SoC[1] designs where you are merging a digital platform design with a custom platform design, you might have a mismatch in the database units per user unit (DBUperUU) settings if your different design teams are using different standards. For example:

- Library Exchange Format (LEF) and Design Exchange Format (DEF) designs in SoCE[2] typically use a DBUperUU of `2000`.
- Virtuoso Design Environment custom designs typically use a DBUperUU of `1000`.

You need to scale the custom design data to `2000` DBUperUU so that the digital library elements and custom design data can be used together in a mixed-platform design environment. You will want to scale all of your design and reference libraries including the PDK library.

To correct the mismatch in database units, you must do the following:

1. Change the DBUperUU in the technology file.

   See "Changing the DBUperUU in the Technology File" on page 306.

2. Use `XScale` to scale the physical elements to compensate for the change in DBUperUU.

   See "Using XScale to Scale Design Data" on page 306.

1. System-on-Chip
2. System-on-Chip Encounter

Changing the DBUperUU in the Technology File

To change the DBUperUU in the technology file, do the following:

1. Write the PDK technology file to an ASCII file.
2. Change the DBUperUU setting in the ASCII technology file to the desired value (`2000` DBUperUU for this example).
3. Load the modified ASCII technology file back into the PDK library.
4. Save.

Important

Do not use the `techSetDBUPerUU` function to set DBUPerUU, because the technology file rules and devices will not be updated properly.

Using XScale to Scale Design Data

To scale the libraries and adjust for the change in DBUperUU, use `XScale` as follows:

```
XScale -mag magFactor
       -lib libName
       [-cell cellName]
       {-view viewName | -viewType viewTypeName}
       [-loadFile skillFileName]
       [-doNotScaleCellFile fileName]
```

For example, to scale the physical elements only (rather than everything in the library), you can specify the `layout` view in your library as follows:

```
XScale -mag 2 -lib myLib -view layout
```

The arguments for `XScale` are as follows:

`-mag` *`magFactor`*	Magnification factor in positive integer or float. For values greater than one, the scale is up. For values less than one, the scale is down. Magnification factor applies to all cellviews getting processed.
`-lib` *`libName`*	Name of the design library that is to be processed. You can specify only one library at a time.

`-cell` *`cellName`*	(Optional) Name of the cellview that is to be processed. This is an optional argument. If no cell name is specified, XScale processes all the cells in the library.

Important

Use either `-view` or `-viewType`: You cannot use both options together.

`-view` *`viewName`*	Name of the view that is to be processed. If a cell name is not specified, the views with the given view name in all cells in the library are processed.
`-viewType` *`viewTypeName`*	Type of cellviews that are to be processed. If a cell is specified, cellviews of the given type in the cell are processed. Otherwise, cellviews of the given type in the library are processed.
`-loadFile` *`skillFileName`*	SKILL file to be loaded.
`-doNotScaleCellFile` *`fileName`*	Control file to be loaded.

Notes:

- `XScale -h` returns the command line syntax.
- The library definitions file in the directory where you run `XScale` must contain any libraries you want to process.

  Typically, you run `XScale` from the design directory where you start Cadence software.
- You might need to reopen a design to see updates to cellview bounding boxes.
- To scale all instances, you can specify the instance master as an input argument to `XScale`.
- `XScale` does not support objects such as markers, rows, boundaries, and blockages.
- ROD alignments and MPP/MPR data is also scaled when the `XScale` command is executed.

A

Bindkeys and Access Keys

This appendix covers the following sections related to bindkeys and access keys:

An Introduction to Bindkeys and Access Keys

A bindkey, or *accelerator*, is a keyboard key or mouse button that is linked (bound) to a Cadence SKILL language command or function name. When you press the key, key sequence, or mouse button, the command executes.

A menu access key is an underlined letter that maps to pressing the *Alt* key plus the letter to post a menu or choose a menu item.

Important

Due to the use of the *Alt* key modifier for Access Keys, it is recommended that you avoid using this key when setting up your bindkeys.

You assign bindkeys per application. For each application, a bindkey can have two settings: a normal setting and a user entry function (also called an "enterfunction") setting. For both normal settings and user entry function settings, you can bind a key or button to one command or a string of space-separated commands. Cadence supplies default files for binding keys in *your_install_dir*`/tools/dfII/samples/local`.

Default Bindkey Files for Specific Applications

Application	File
Virtuoso layout suite	`leBindKeys.il`
Place-and-route	`prBindKeys.il`
Schematic entry	`schBindKeys.il`
Layout and schematic entry	`leSchBindkeys.il`

You can change the defaults via the bindkey user interface (using the Bindkey Editor form, selectable from the CIW (*Options – Bindkeys*)) or directly in your `.cdsinit` file.

Note: If you change your `.cdsinit` search mechanism, so as to load a different `.cdsinit` file, the defaults specified in the initial `.cdsinit` file may not be there. For more information see Setting the SKILL Search Path.

Important

Support of Xt/Motif will be removed from a future Virtuoso release. Resultantly, the use of OSF virtual key names will no longer be possible, as that facility is specific to Xt/Motif.

Viewing the Current Bindkeys for an Application

Note: See also Configuring Application Bindkeys.

To view current bindkey settings for an application, do the following:

1. From the Command Interpreter Window (CIW), choose *Options – Bindkeys*.

   The Bindkey Editor form is displayed.

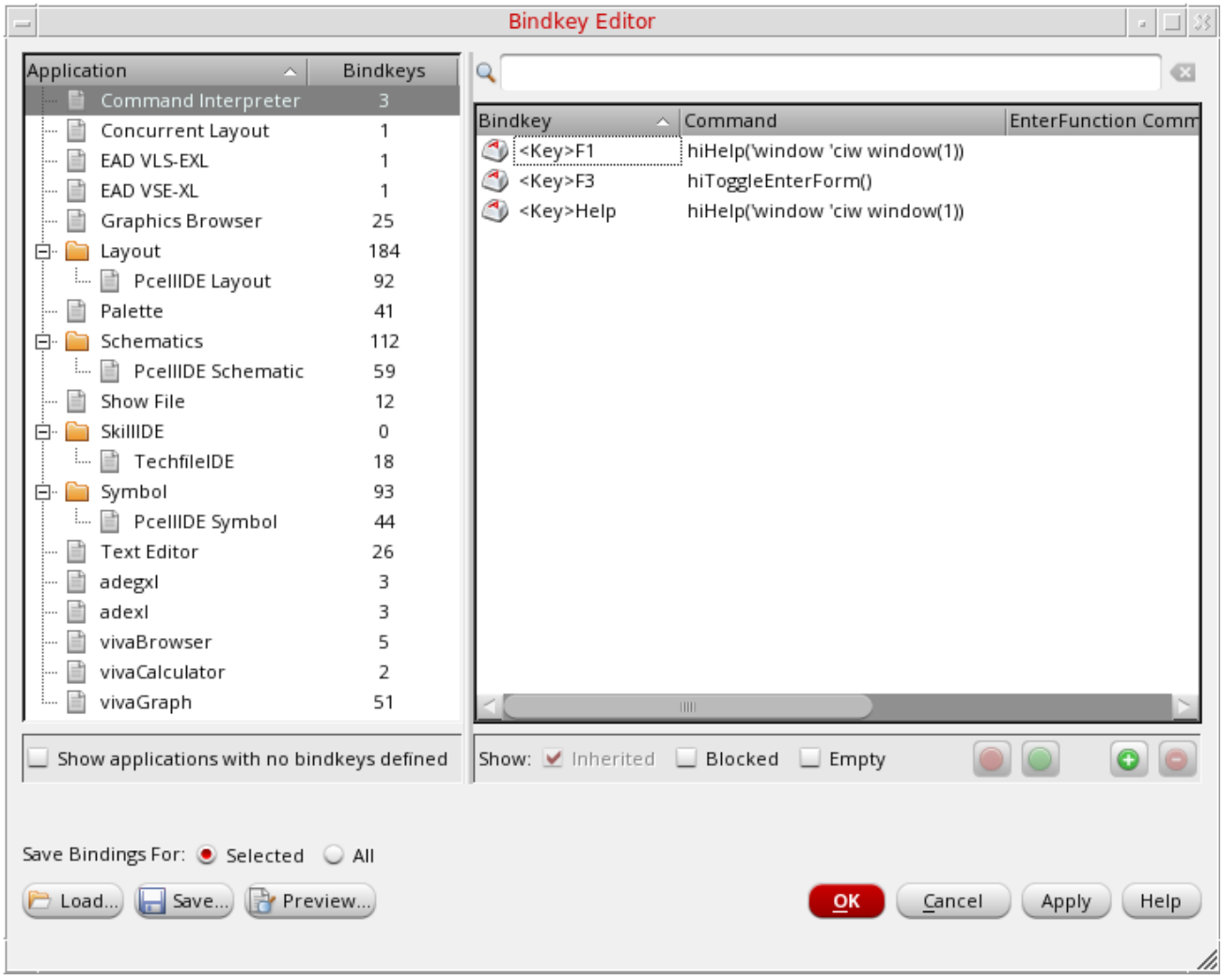


Figure A-1 Bindkey Editor Form

Note: The *Application Tree*, to the left of the form will, by default, initially display all applications for which bindkeys have been defined.

2. In the *Application Tree*, select the application whose bindkeys you want to view.

The *BindKey Table*, to the right of the form, will update to list those bindkeys associated with the currently selected application.

3. Optionally, use the *Search* field to filter the bindkey listing.
4. To view bindkey settings for the currently selected application, either:
   - Scroll through the listed bindkeys in the *BindKey Table* to view the current settings for one or more bindkeys.

     or

   - Select the *Preview* button to display a listing of **all** of the bindkeys associated with the selected application in a text window.

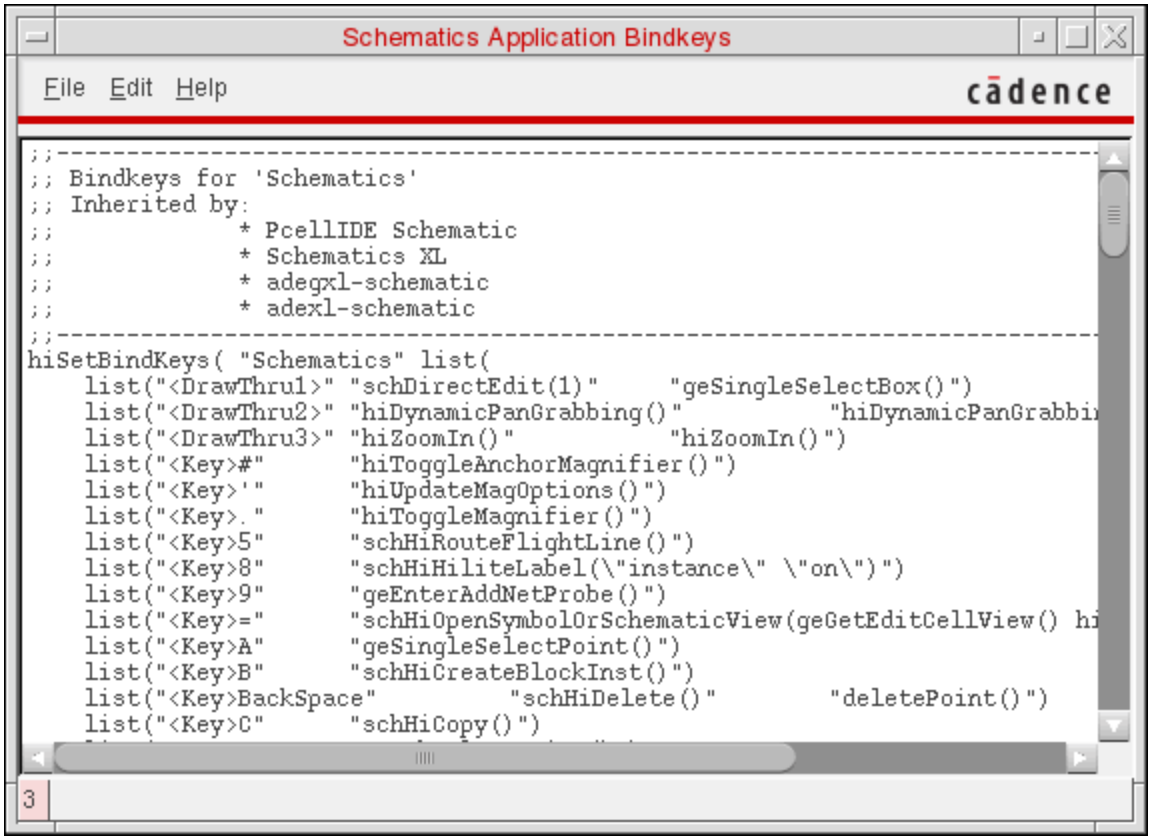


Figure A-2 Current Bindkey Settings for the Schematics Application

Restrictions to Setting Bindkeys

The following restrictions apply to setting bindkeys:

- You must not use the *Alt* key as a modifier for any printable character (see `hiBindkey` in the *Cadence User Interface SKILL Reference* for more information).
- You must not assign alphanumeric bindkeys (a–z, A–Z, or 0–9) to the CIW or to an encapsulation window. (An encapsulation window is the user interface to software Cadence has modified to run in the Virtuoso Design Environment.)
- You must not use the left or middle mouse buttons for setting bindings in the CIW or in an encapsulation window. Using these buttons to set bindings will conflict with their normal actions.
- You must not assign anything to the *F3* or *F4* function key. Cadence software uses the *F3* key to display options forms and the *F4* key for partial selection in graphics windows.
- You must not assign anything to the *Lock* (*Caps Lock*), *!* (exclamation point), and ~ (tilde) keys.
- You must also not assign anything to the following keys:

Key	Current Usage
F1	Used for Help
F11	Reserved
F12	Reserved

Configuring Application Bindkeys

Important

You can override bindkeys inherited from the root application by defining a new bindkey for an inherited application.

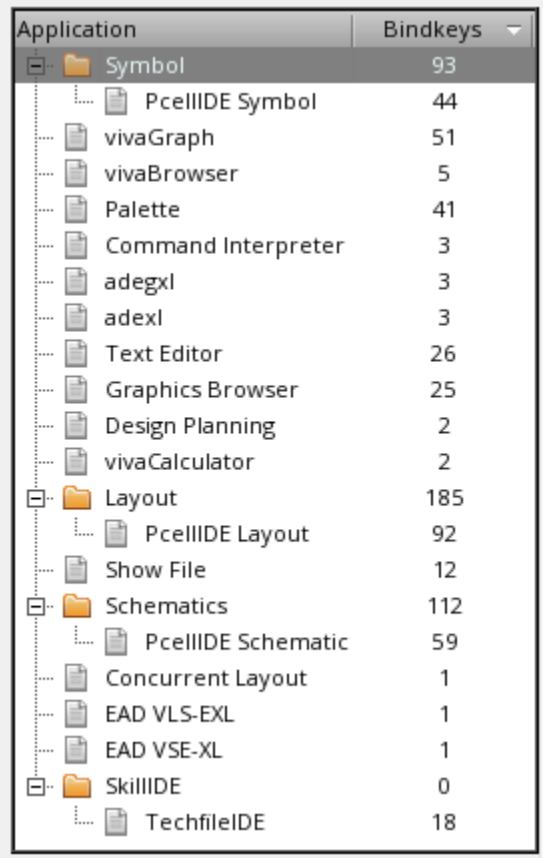


Figure A-3 Inherited Applications Shown for Schematics

You can configure (add, edit, or delete) the bindkey settings for an application using the Bindkey Editor form that is accessible by selecting *Options – Bindkeys* from the CIW.

Note: See also `hiConfigureBindKeys` in the *Cadence User Interface SKILL Reference*.

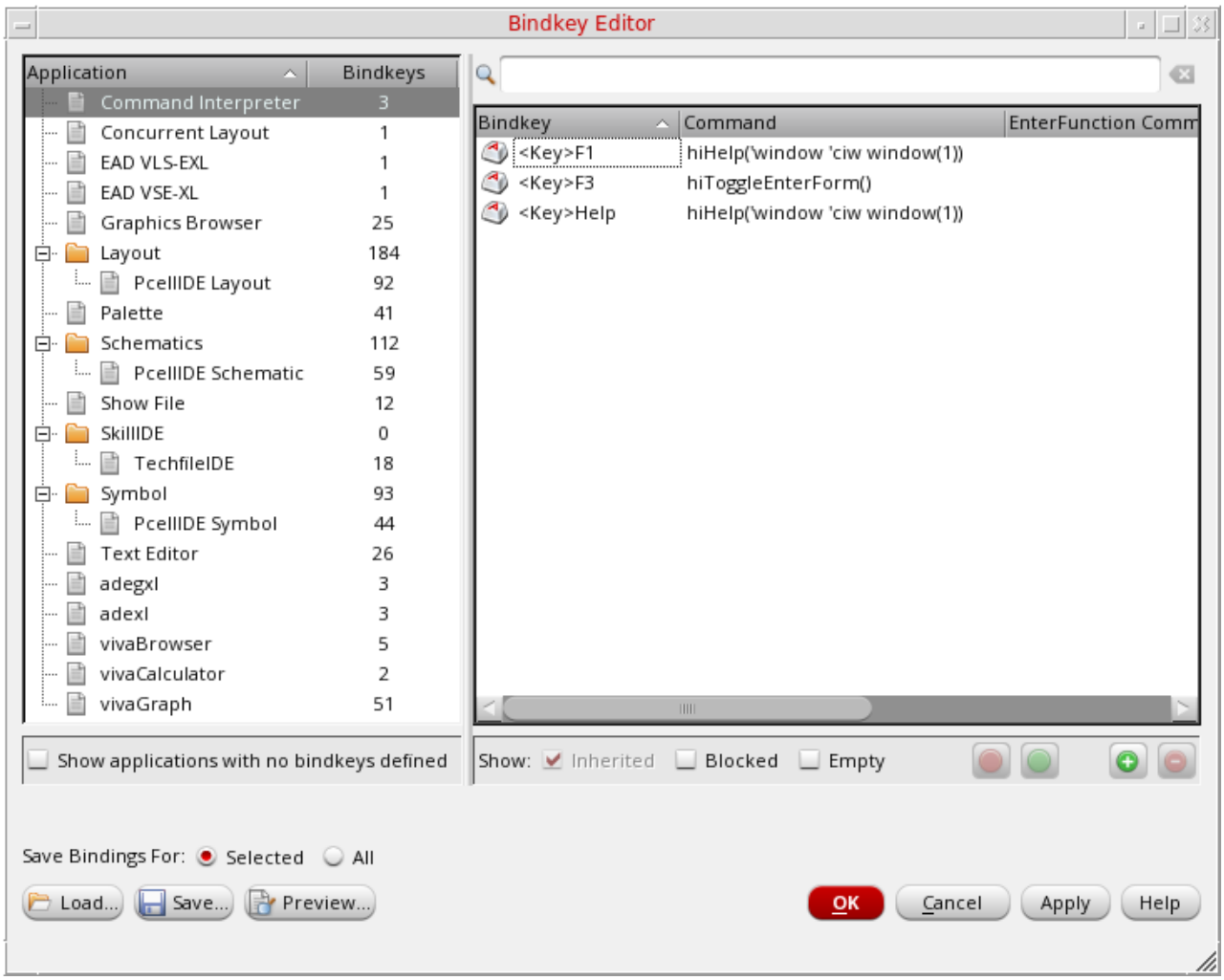


Figure A-4 The Bindkey Editor Form

The Bindkey Editor form comprises of the following options:

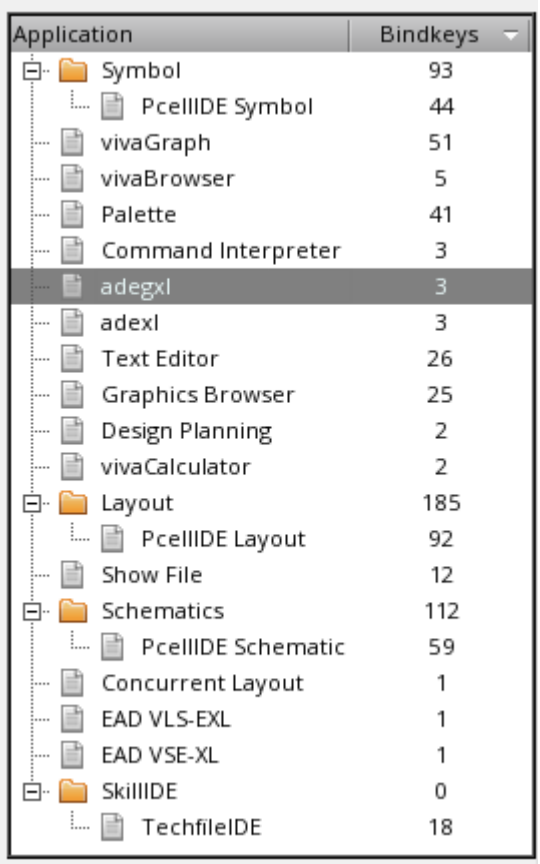
Option	Description
Application Tree	

Figure A-5 Selecting an Application to View Its Bindkeys

The *Application Tree* displays those applications for which bindkeys are currently defined. Additionally, by checking the *Show applications with no bindkeys defined* checkbox, all Virtuoso applications, irrespective of whether bindkeys exist for them or not, will also be displayed. Selecting an *Application* will update the *Bindkey Table* to show the bindkeys associated with that application.

You can select more than one application simultaneously and the *Bindkey Table* will update to display the bindkeys for all selected applications. From there, you can then select the bindkey settings that you want to view or edit.

Show applications with no bindkeys defined

Select to also show those applications that do not currently have their bindkeys defined.

Note: These applications will be shown collectively alongside those applications that already have their bindkeys defined.

Save Bindings For:

Select which applications will be taken into consideration by the Preview and Save operations.

Preview

Select to display a text window detailing the bindkeys.

See also Viewing the Current Bindkeys for an Application.

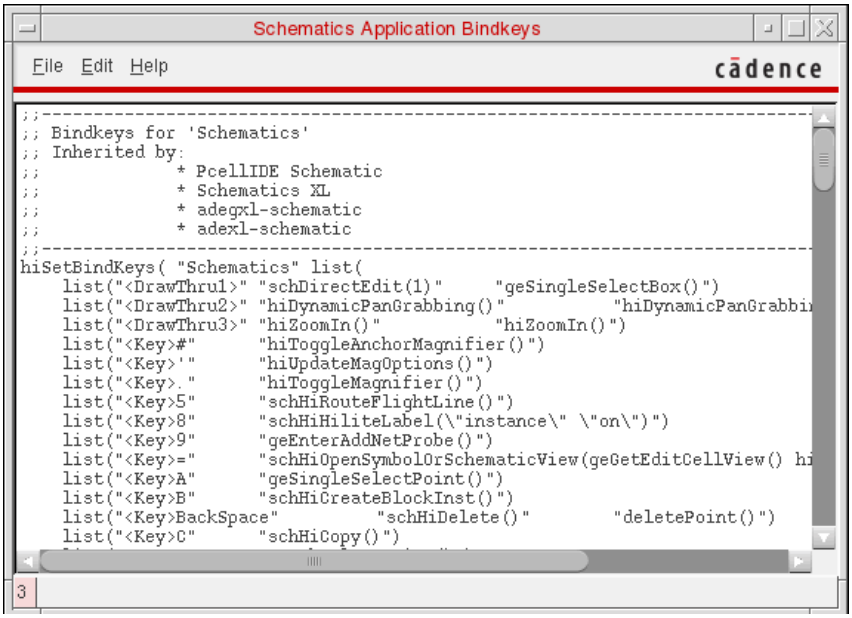


Figure A-6 Bindkey Listing

Note: Applications which inherit the bindkeys for the currently-selected application are also listed.

Load

Select to display the Load Bindkeys form where you can load previously saved bindkeys for use.

Save

Select to display the Save Bindkeys form where you can save bindkey settings.

Important

Bindkeys are not autoloaded. You must explicitly load any saved bindkey files from a file which is automatically loaded, such as the `.cdsinit` file.

Search

Enter search criteria to filter the bindkeys that are listed in the *Bindkey Table*.

Tip

You can use the *Search* field to group bindkeys by the same base key. For example, if you wanted to show all bindkeys that are associated with the `A` key, enter `<KEY>A` in the *Search* field. This will then display all the associated `A` key bindkeys, for example `Shift`, `Ctrl`, and `Ctrl+Shift`.

Bindkey Table

Lists all of the current bindkeys associated with the currently selected application(s) in the *Application Tree*, including their related *Command* and *EnterFunction Command*.

- Bindkey

The current key or mouse binding for the selected bindkey.

See also Restrictions to Setting Bindkeys and Format for Mouse and Key Bindings.

- Command

Displays the SKILL function associated with the currently selected bindkey.

This field can be edited by double clicking in the table cell.

For example, if you wanted to set a command to open the Library Manager, type the following:

```
ddsOpenLibManager()
```

If you want to bind more than one function, separate them with a space. For example, to open the Library Manager and the Library Path Editor, type `ddsOpenLibManager() ddsHiOpenCdsLibEditor()`.

See also How to Determine What to Type in the Command Field.

- *EnterFunction Command*	Displays the user entry function (also called an “enterfunction”) that is currently associated with the selected bindkey. This field can be edited by double clicking in the table cell. If you bind more than one function, again, separate them with a space. A given bindkey can have a single function, or set of functions, assigned while the software is in its normal, “ready” state. You can however also assign a different function, or set of functions, while the software is entering a command. This is called “enterfunction” mode. The *EnterFunction Command* option is used to specify this difference. Using the Bindkey Editor form, rather than `hiSetBindKeys()`, differs in use for enterfunction commands. If you call `hiSetBindKeys()` and only specify a regular command, and no enterfunction command is specified, that command will work both in regular mode and enterfunction. With the form, if you do not specify an enterfunction command, it will create an “empty” enterfunction command.
Show	Lets you show or hide the following types of bindkeys: ■ *Inherited* ■ *Blocked* ■ *Empty* Based on your selection, Bindkey Editor appropriately highlights bindkeys. Bindkeys that have been overridden, using `nil`, in a child application are represented in the child application using strikethrough text. By default, the *Inherited* option is selected to list bindings that are inherited from a root application and are also shown, with a gray background, when a child application is selected.
Block Binding and *Unblock Binding*	Lets you block or unblock binding the selected bindkey. Use these options to prevent or allow propagation to the parent application. These options are available for bindkeys of a child application.

Application

Shows the application that the bindkey is associated with if more than one application is selected in the *Application Tree*. Otherwise, the column will not be displayed.

Displaying the related application is especially useful in the following situations:

- When a child application is selected and *Show inherited bindings* is checked.

  Here, multiple applications can be listed in the *BindKey Table*, and clicking on the *Application* column header will sort by application; distinguishing inherited bindings from those bindings defined for the current application.

- When multiple applications are selected in the *Application Tree*.

  Here, for example, you may now want to use the *BindKey Table* to view bindings for `Ctrl<Key>X` across all applications (to achieve this, `shift-select` the applications in the *Application Tree*, then use *Search* to filter the bindings). Without the *Application* column you would not know which application a bindkey was inherited from.

Select the *Add binding* button to create a new bindkey.

Select the *Remove binding* button to remove an existing (customized) bindkey.

For more information, see Adding a Bindkey, Deleting a Bindkey, and Editing a Bindkey.

Adding a Bindkey

1. In the *Application Tree*, to the left of the form, select the application that you want to create a new bindkey for.
2. Select the *Add binding* button to add the new bindkey.

   You will now be prompted to "*Press the new bindkey combination*".

Figure A-7 Specifying a New Bindkey for the Schematics Application

Note: If you have more than one application selected in the *Application Tree*, an additional *Application* column will be added to the table, specifying what application a bindkey belongs to.

3. To add the new bindkey, you can either type the required bindkey directly into the *Bindkey Table* or select the appropriate *Mouse* or *Key* icons to invoke the Capture mouse binding and Capture key binding forms.

   Once you have specified the bindkeys, the new bindkey entry will be recorded in the *Bindkey Table*.

4. Optionally, add a *Command* and *EnterFunction Command* to be associated with the new bindkey by double-clicking on the appropriate cell in the *Bindkey Table*.

   Note: When you are adding a new bindkey, the bindkey will be displayed in blue text in

the *Bindkey Table* indicating that is has been newly created, but not yet saved.

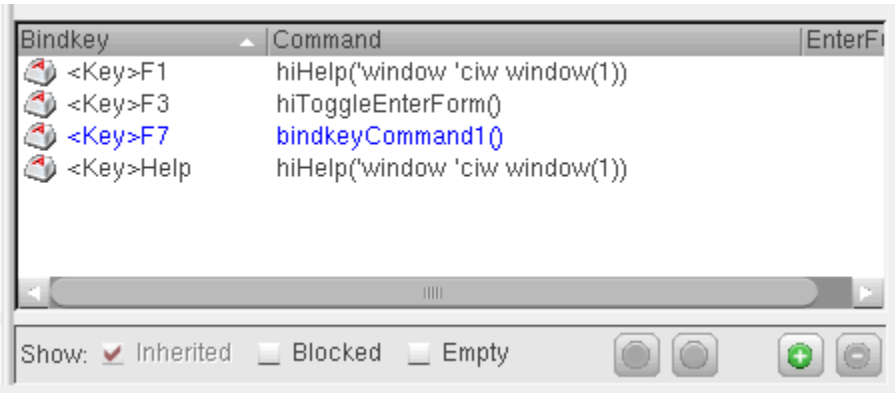


Figure A-8 A New Bindkey Added (but not yet Saved)

5. Click the *Save* button.

   The Save Bindkeys form is displayed where you can now save the application bindkeys to file for reuse.

Capturing Mouse and Key Bindings

You can use the Capture mouse binding and Capture key binding forms to capture actions that cannot be intuitively capture using the text entry field.

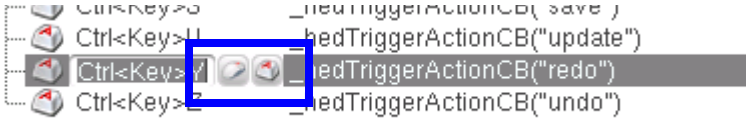


Figure A-9 Mouse and Key Binding Icons

Selecting the *Mouse* icon displays the Capture mouse binding form while selecting the *Key* icon displays the Capture key binding form.

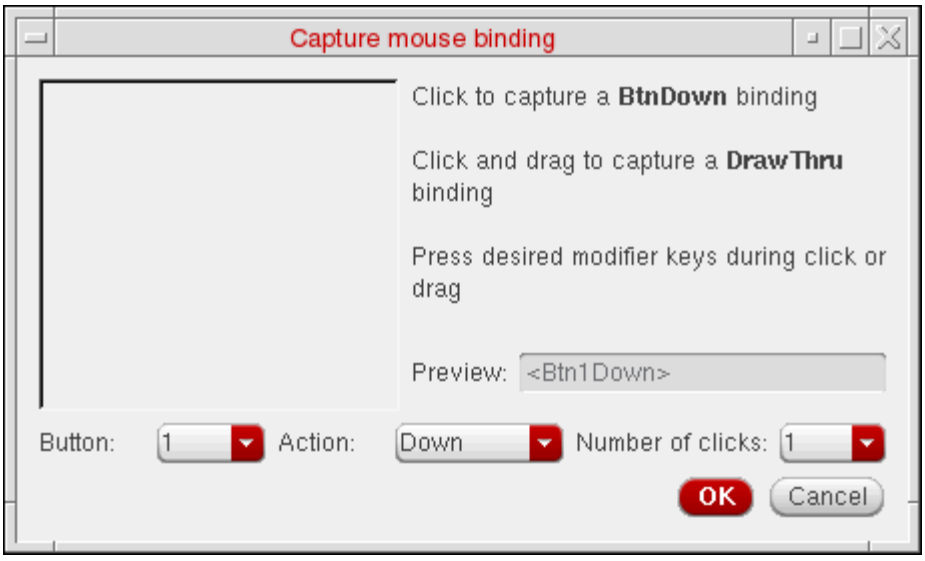


Figure A-10 Capture mouse binding form

Note: Perform the mouse binding to be captured within the blank window frame area in the left of the form.

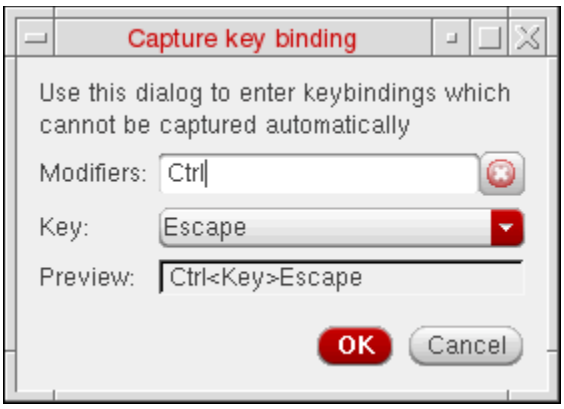


Figure A-11 Capture key binding form

Duplicate Bindkeys

If, when creating a new bindkey, you enter a bindkey combination that already exists, a *Duplicate bindkeys detected* message box will be displayed.

Figure A-12 Duplicate Bindkey Detected

From here, you can choose to *Discard* the new bindkey setting, or to *Apply* the bindkey setting anyway.

If you chose to *Apply* the bindkey, the *Bindkey table* will now show that there is a duplicate bindkey issue.

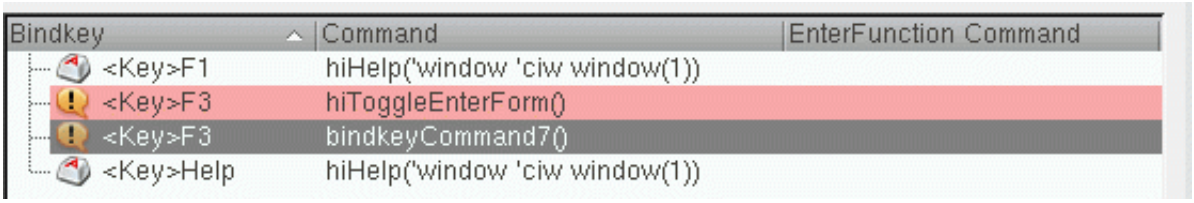


Figure A-13 Duplicate Use of F3 Bindkey

From here however, you can now choose to delete one of the keybindings commands, be that the original binding or the newly created one.

Deleting a Bindkey

1. Select *Options – Bindkeys* from the CIW window.

   The Bindkey Editor form is displayed.

2. In the *Application Tree*, to the left of the form, select the application that you want to remove a new bindkey from.

3. In the *Bindkey Table*, to the right of the form, select the bindkey that you want to delete.

   Note: If required, you can select multiple bindkeys to be deleted.

4. Click the *Remove binding* button.

   The bindkey will be deleted from the currently selected application.

5. Click the *Save* button.

   The Save Bindkeys form is displayed where you can now save the application bindkey list.

Note: When a bindkey is deleted from the *Bindkey Editor*, rather than it being removed completely, an empty bindkey is left in its place. So, for example, you cannot delete a `Ctrl+E` binding, rather it becomes an empty (unused) binding. Such bindings cannot be viewed in the *Bindkey Editor*, rather, the saved bindkey file will store any empty settings (which can also be viewed when performing a *Preview*).

Editing a Bindkey

1. Select *Options – Bindkeys* from the CIW window.

   The Bindkey Editor form is displayed.

2. In the *Application Tree*, select the application(s) that you want to edit bindkeys in.

   The *BindKey Table* will update to show all of the bindkeys in the currently selected application(s).

3. Optionally, use the *Search* option to filter the bindkey listing displayed in the *BindKey Table*.

4. Select the bindkey that you want to edit.

5. Edit the *Command* and *EnterFunction Command* settings as required.

   Double-clicking on the appropriate bindkey option field (*Command* or *EnterFunction Command*) will allow you to edit the bindkey directly in the *BindKey Table*.

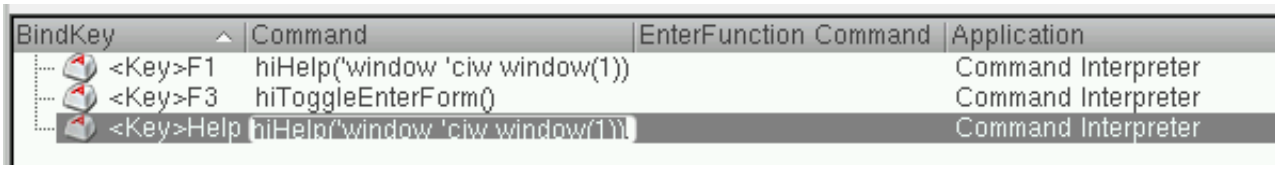


Figure A-14 Direct Edit of a Bindkey in the BindKey Table

6. Click *Apply* to set the bindkey edits.

7. Click *Save* to record the bindkey updates that have been made.

   The bindkey edits can now be used as required.

Format for Key and Mouse Bindings

The following format is used for key and mouse binding modifiers:

```
[modifiers]<operator>detail[(occurrences)] [EF]
```

where:

modifier — (Optional) Specifies the modification key that will be used with the bound key(s) or mouse button
Valid Values:

`Alt`	*Alt* key See also `hiBindkeyModifers` in the *Cadence User Interface SKILL Reference* for important information.
`Shift`	*Shift* key
`Ctrl`	*Ctrl* key
`Meta`	*Meta* key (on Sun keyboards, a black diamond)
`None`	No extra key
`Super`	This is a custom modifier that may not always work with mouse bindings. A `Super` (or `Hyper`) modifier can be set through the use of `xmodmap`, where, for example, you can re-define the use of the ride-sided `Alt` key (`Alt_R`) for use as a `Super` or `Hyper` modifier and assign that to an unused modifier bit. **Note:** After the `Super` or `Hyper` modifiers are defined, they will also be listed by `hiBindKeyModifiers`.
`Hyper`	As above for `Super`.

Note: Not specifying a modifier is the name as specifying `None`.

operator — Specifies the bound key or mouse button
Valid Values:

Note: Surrounding angle brackets (<>) are required.

`Key`	Keyboard key

`Btn1Down`	Left mouse button
`Btn2Down`	Middle mouse button
`Btn3Down`	Right mouse button
`Btn4Down`	Scroll wheel up
`Btn5Down`	Scroll wheel down
`Btn8Down`	Thumb button closest to the wrist
`Btn9Down`	Thumb button farthest from the wrist
`DrawThru1`	Click and hold left mouse button while moving mouse
`DrawThru2`	[same for middle mouse button]
`DrawThru3`	[same for right mouse button]
`DrawThru8`	[same for closest thumb button]
`DrawThru9`	[same for farthest thumb button]

Note: A number of hard-coded keys have been set to be recognized for pan and zoom when performing a `DrawThru` command.

For example, if you are performing a “drag-zoom” (typically using the right-mouse button) or a “drag-select” (typically using the left-mouse button), you can use the Arrow keys and/or `Z` and `Shift+Z` to pan and zoom during that operation.

This is achieved where:

- The Arrow keys will pan in the appropriate direction by by calling `hiAbsolutePan()`
- `Z` will call `hiZoomInAtMouse()`
- `Shift+Z` will call `hiZoomOutAtMouse()`

These are however all hard-coded, not editable through the bindkey interface, and are in addition to the `Esc` key, which calls `cancelEnterFun()` during the `Drawthru` command.

	■ Apart from this, you can also use the `f` key to zoom-fit a design. The `f` key calls the `hiZoomAbsoluteScale()` function.
detail	Specifies the name of the keyboard key, such as `A`, `B`, `F3`, `Tab`, or `Del`
occurrences	(Optional) Specifies the number of times to use a mouse button. Should not be specified for `Btn4Down` or `Btn5Down`. **Note:** Use `2` for double-click, `1` for single-click. If you do not specify a value for occurrences, the software uses `1` (single-click). If you do specify a value, the surrounding parentheses are required.
`EnterFunction Command`	(Optional) Indicates that the binding is effective during a user entry function (such as drawing a rectangle).

Additional Guidelines for Non-ASCII-7 Character Bindkey Definitions

Bindings may also be defined for keys that generate characters that are not represented in the `ASCII-7` character set, but are part of the `ASCII`-8 set.

Note: These characters are also referred to as "extended ASCII characters".

Such extended ASCII characters are available on non-US English keyboards, and have numeric values greater than 127, which are not currently parseable by SKILL (such as the upside-down exclamation point "`exclamdown`", the cent sign, the pound sterling sign, the uppercase and lowercase o with `"umlaut"`, the uppercase and lowercase e with a grave accent, and so on).

To define these keys, the name of the character must be used, rather than the character itself.

The full list of these character names is as follows (listed in numerical value order):

```
nobreakspace, exclamdown, cent, sterling, currency, yen, brokenbar, section,
diaeresis, copyright, ordfeminine, guillemotleft, notsign, hyphen, registered,
macron, degree, plusminus, twosuperior, threesuperior, acute, mu, paragraph,
periodcentered, cedilla, onesuperior, masculine, guillemotright, onequarter,
onehalf, threequarters, questiondown, Agrave, Aacute, Acircumflex, Atilde,
Adiaeresis, Aring, AE, Ccedilla, Egrave, Eacute, Ecircumflex, Ediaeresis, Igrave,
Iacute, Icircumflex, Idiaeresis, ETH, Ntilde, Ograve, Oacute, Ocircumflex, Otilde,
Odiaeresis, multiply, Ooblique, Oslash (same as Ooblique), Ugrave, Uacute,
Ucircumflex, Udiaeresis, Yacute, THORN, Thorn, ssharp, agrave, aacute, acircumflex,
atilde, adiaeresis, aring, ae, ccedilla, egrave, eacute, ecircumflex, ediaeresis,
igrave, iacute, icircumflex, idiaeresis, eth, ntilde, ograve, oacute, ocircumflex,
otilde, odiaeresis, division, oslash, ooblique (same as oslash), ugrave, uacute,
ucircumflex, udiaeresis, yacute, thorn, ydiaeresis
```

Note: These names are case-sensitive.

Additionally, the names of regular ASCII symbols can be used in place of the symbol (for `space`, the name must be used):

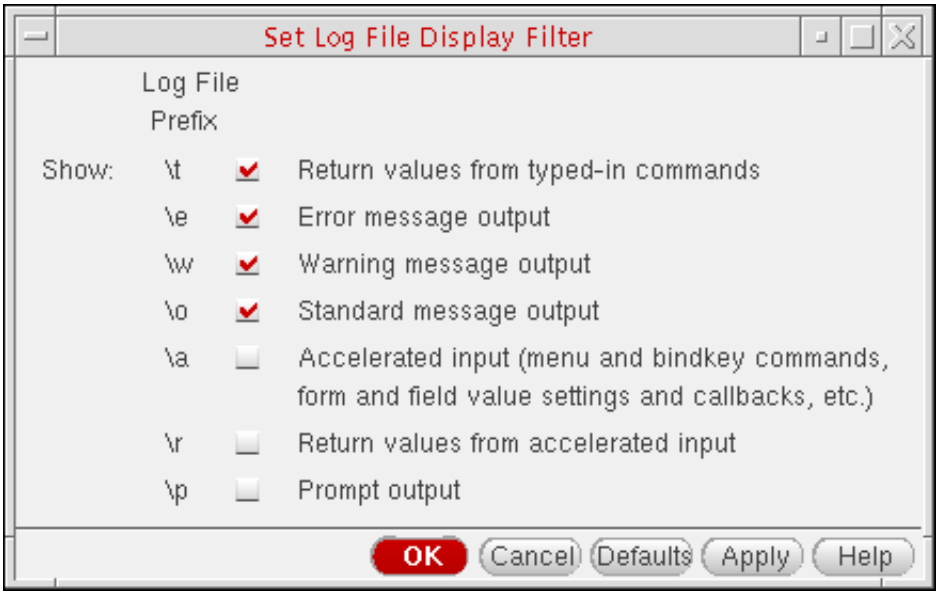
```
' ' = "space"
! = "exclam"
" = "quotedbl"
# = "numbersign"
$ = "dollar"
% = "percent"
& = "ampersand"
' = "apostrophe"
' = "quoteright" // deprecated
( = "parenleft"
) = "parenright"
* = "asterisk"
+ = "plus"
, = "comma"
- = "minus"
. = "period"
/ = "slash"
: = "colon"
; = "semicolon"
< = "less"
= = "equal"
> = "greater"
? = "question"
@ = "at"
[ = "bracketleft"
\ = "backslash"
] = "bracketright"
^ = "asciicircum" // a/k/a "circumflex"
_ = "underscore"
` = "grave"
` = "quoteleft" // deprecated
{ = "braceleft"
| = "bar"
} = "braceright"
~ = "asciitilde"
```

How to Determine What to Type in the Command Field

To determine the SKILL function associated with a menu command, do the following:

1. From the CIW, choose *Options – Log Filter*.

   The Set Log File Display Filter form appears.



2. Check the *Accelerated input* check box.
3. Click *OK*.

   SKILL functions appear in the CIW output area.
4. Click the menu command whose SKILL function you want to know.

   The SKILL function associated with the menu command appears in the CIW. This function will also be the last one in your log file. The default log file name is `CDS.log` in your home directory (unless you specified otherwise).

Setting Bindkeys Across Sessions Using the .cdsinit File

You can use the same key or mouse bindings for every session by adding a SKILL `hiSetBindKey` command to your `.cdsinit` file. as follows:

```
hiSetBindKey("t_application" "t_key" "t_SKILL_cmd")
```

where

t_application	Specifies the name of the application or window for which you want to set a bindkey
t_key	Specifies the name of a key, key sequence, or mouse button
t_SKILL_cmd	Specifies a SKILL command

Here are some examples.

- To open the Library Manager when you press `F6` in the CIW, type the following command in your `.cdsinit` file:

  ```
  hiSetBindKey("Command Interpreter" "<Key>F6" "ddsOpenLibManager()")
  ```

- To open the Library Manager when you press `F6` in the schematic editor, type the following in your `.cdsinit` file:

  ```
  hiSetBindKey("Schematics" "<Key>F6" "ddsOpenLibManager()")
  ```

Note: The best method to set bindkeys persistently is to create a file with the bindkey settings in it and load this file from the `.cdsinit` file.

See also Restrictions to Setting Bindkeys.

Controlling the Mouse Bindings Display in the CIW

The mouse bindings line of the CIW displays the keys bound to the mouse buttons for the current command or window.

- On the CIW input line, type the following to turn the CIW bindkey display off:

  ```
  hiGetCIWindow( )->displayMouseBinding=nil
  ```

- On the CIW input line, type the following to turn the CIW bindkey display on in the CIW:

  ```
  hiGetCIWindow( )->displayMouseBinding=t
  ```

Controlling Bindkey Display on Menus

The following bindkeys, when assigned to menu commands, appear to the right of the command on the menu:

- Alphanumeric keys: A–Z, 0–9
- Function keys: F1–F35
- Named keys (such as `Return`, `Escape`, `Delete`, `BackSpace`, `Space`, `Tab`, `Up`, keypad keys, and keys marked `R` and `L`)

  ,`Escape` and `Delete`, and `BackSpace` are abbreviated in the menu display as *Esc* and *Del*, but they must be spelled out with correct use of uppercase and lowercase in the key definition.
- Punctuation and symbol keys
- Any of the above keys in combination with the modifiers `Alt`, `Shift`, `Ctrl`, and `Meta`

These bindkeys are not displayed on menus:

- Mouse bindings
- Bindings active during a user-entry function.

The menus show both default bindkeys and keys you assign. The CIW has no default bindkeys of the type that appear on menus. You do not see any bindkeys next to CIW commands unless you assign them.

➤ On the CIW input line, type the following command to turn the menu bindkey display off:

```
hiGetCIWindow( )->displayMenuBindkeys=nil
```

Bindkey assignments do not appear unless the SKILL function you assign to the key **matches** the SKILL function (including blank spaces and capitalization) used by the menu command. For example, if the menu uses

```
hiGetBannerMenus( getCurrentWindow())
```

(notice the space between the first parenthesis and the `getCurrentWindow` function) and you bind a key to

```
hiGetBannerMenus(getCurrentWindow())
```

(without the space) the binding works, but the bindkey assignment does not appear.

Note: If several bindkey combinations are bound to a menu command, the combination requiring the least number of characters is the one displayed. If several combinations using the same number of characters are bound to a command, the most recent one is displayed.

Here are some common assignments:

Bindkey Assignment	What it represents	Menu appearance
`<Key>a`	Unshifted letter A	A
`<Key>Escape`	Escape key	Esc
`<Key>Tab`	Tab key	Tab
`<Key>BackSpace`	BackSpace key	BackSpace
`<Key>Delete`	Delete key	Del
`<Key>Return`	Return key	Return
`<Key>F3`	Function key F3	F3
`<Key>1`	Numeric key 1	1
`Shift<Key>a`	Capital letter A	Shift+A
`Shift<Key>F3`	Shift key + function key F3	Shift+F3
`Shift<Key>Escape`	Shift key + Escape key	Shift+Esc
`Ctrl<Key>a`	Control key + lowercase a	Ctrl+A
`Ctrl<Key>F3`	Control key + function key F3	Ctrl+F3
`Ctrl<Key>Escape`	Control key + Escape key	Ctrl+Esc
`Shift Ctrl<Key>a`	Shift key + Control key + A	Shift+Ctrl+A
`Shift Ctrl<Key>F3`	Shift key + Control key + F3	Shift+Ctrl+F3
`Shift Ctrl<Key>Escape`	Shift key + Control key+ Escape key	Shift+Ctrl+Esc
`Meta<Key>m`	Meta key + unshifted M	Meta+M

Default Keybindings for Text Fields

Virtuoso IC6.x uses a Qt-based UI. As part of the GUI system, the following default bindings for text fields are used:

Key Sequence	Action
`Left Arrow`	Moves the cursor one character to the left.
`Shift+Left Arrow`	Moves and selects text one character to the left.
`Right Arrow`	Moves the cursor one character to the right.
`Shift+Right Arrow`	Moves and selects text one character to the right.
`Home`	Moves the cursor to the beginning of the line.
`End`	Moves the cursor to the end of the line.
`Backspace`	Deletes the character to the left of the cursor.
`Ctrl+Backspace`	Deletes the word to the left of the cursor.
`Delete`	Deletes the character to the right of the cursor.
`Ctrl+Delete`	Deletes the word to the right of the cursor.
`Ctrl+A`	Selects all.
`Ctrl+C / Ctrl+Insert`	Copies the selected text to the clipboard.
`Ctrl+K`	Deletes to the end of the line.
`Ctrl+V / Shift+Insert`	Pastes the clipboard text into line edit.
`Ctrl+X / Shift+Delete`	Deletes the selected text and copies it to the clipboard.
`Ctrl+Z`	Undoes the last operation.
`Ctrl+Y`	Redoes the last undone operation.

Default Keybindings for CIW

You can use the following default keybindings in the CIW edit area (edit area is defined as the area below the last light-gray background command block in the CIW):

Key Sequence	Action
`Esc`	If the the text is selected and the `Esc` key is pressed, it clears the selection. However, if no text is selected, the `Esc` key clears the text from the input area. In addition, the `Esc` key closes the regular forms and hides the option forms, which are displayed by pressing the `F3` key, when the focus is on them. In case of option form, the `Esc` key only hides the form when pressed once and does not cancel the command. However, if the `Esc` key is pressed twice, the first press hides the form and the second press cancels the command.
`Ctrl+Esc` `Ctrl+C`	A parser error will be generated.
`Ctrl+Insert` `Ctrl+C`	Copies selected text to clipboard when text is selected in the input or output areas. **Note:** Use `F16`, the copy key, on Sun keyboards.
`Delete` `Ctrl+D`	Deletes the next character.
`Meta+Delete` `Meta+D`	Deletes the next word.
`Backspace` `Ctrl+H`	Deletes the previous character
`Meta+Backspace` `Meta+H`	Deletes the previous word

`Delete`
`Backspace`
`Ctrl+W`
	Deletes the selected text

`Ctrl+Backspace` `Ctrl+U`	Deletes until the beginning of a line

`Ctrl+Delete` `Ctrl+K`	Deletes until the end of a line
`Down Arrow`	Displays next history command if the History In Place option is selected and the cursor is at the last line of the edit area
`Up Arrow`	Displays the previous history command if the *History In Place* option is selected and the cursor is at the end or at the first line of edit area

`Shift+Enter`
`Shift+Return`
`Ctrl+J`
`Ctrl+M`
`Enter`
`Return`
	■ If syntax highlighting is off, executes command or sends partial command to SKILL.
	■ If syntax highlighting is on, `Ctrl+J` and `Ctrl+M` will insert a new line.
	■ If syntax highlighting is on and the *Enter Key Executes Command* option is selected, `Enter` and `Return` will execute the command and `Shift+Enter` and `Shift+Return` will insert a new line, else if the *Enter Key Executes Command* option is not selected, `Enter` and `Return` will insert a new line and `Shift+Enter` and `Shift+Return` will execute the command.
	Note: If syntax highlighting is on, the command will only be sent to the SKILL parser if the syntax is complete, otherwise, the CIW syntax checker will beep and do nothing. If syntax highlighting is off, the command or partial command will be sent to the SKILL parser and the parser will execute the command once it receives a complete command.

`Alt+Enter` `Alt+Return`	Executes the selected commands. If syntax highlighting is on and the text is not syntax complete, the SKILL parser copies the selected text to the edit area and beeps
`Alt+J` `Ctrl+Delete`	Joins next command block (only available if syntax highlighting on)
`Shift+Alt+J` `Ctrl+Backspace`	Joins previous command block (only available if syntax highlighting on) **Note:** Use `Ctrl+Backspace` if the cursor is at the start ofthe command block
`Down Arrow` `Ctrl+N`	Moves the cursor down one line

`Left Arrow` `Ctrl+B`	Moves left one character
`Right Arrow` `Ctrl+F`	Moves right one character
`Alt+<` `Alt+,`	Moves to beginning of current command block
`Alt+>` `Alt+.`	Moves to end of current command block
`Alt+Home`	Moves to the beginning ofthe input area
`Home` `Ctrl+A`	Moves to the beginning of the line
`Alt+End`	Moves to the end of the input area
`End` `Ctrl+E`	Moves to the end of the line
`Ctrl+Q`	Moves to matching parenthesis, brace or bracket (if match is highlighted) (only available if syntax highlighting on)
`Ctrl+Q`	Moves to nearest parenthesis, brace or bracket (if no match is highlighted) (only available if syntax highlighting on)
`Ctrl+Right Arrow` `Meta+F`	Moves the cursor to the next word
`Ctrl+Left Arrow` `Meta+B`	Moves the cursor to the previous word
`Up Arrow` `Ctrl+P`	Moves up one line
`Shift+Down Arrow` `Shift+Ctrl+N`	Selects down one line
`Shift+Ctrl+Q`	Selects expression or extends selected expression (only available if syntax highlighting is on and selection does not cross a command block boundary)
`Shift+Left Arrow` `Shift+Ctrl+B`	Selects left one character

`Shift+Ctrl+Right Arrow` `Shift+Meta+F`	Selects the next word
`Shift+Ctrl+Left Arrow` `Shift+Meta+B`	Selects the previous word
`Shift+Right Arrow` `Shift+Ctrl+F`	Selects right one character
`Shift+Alt+<` `Shift+Alt+,`	Selects to beginning of current command block
`Shift+Alt+Home` `Shift+Ctrl+Home`	Selects to beginning of input area
`Shift+Home` `Shift+Ctrl+A`	Selects up to the beginning of the command line
`Shift+Alt+>` `Shift+Alt+.`	Selects up to the end of the current command block
`Shift+Alt+End` `Shift+Ctrl+End`	Selects up to the end of the input area
`Shift+End` `Shift+Ctrl+E`	Selects up to the end of line
`Shift+Up Arrow` `Shift+Ctrl+P`	Selects up one line
`Shift+Insert` `Ctrl+V`	Paste copied text **Note:** Use the `F18` key on a Sun keyboard
`Middle mouse button`	Pastes selected text
`Ctrl+R`	Redisplays history
`Ctrl+<` `Ctrl+,`	Scrolls output area towards the left
`Ctrl+>` `Ctrl+.`	Scrolls output area towards right
`Shift+Ctrl+<` `Shift+Ctrl+,`	Scrolls output area one page towards the left
`Shift+Ctrl+>` `Shift+Ctrl+.`	Scrolls output area one page towards the right

`Ctrl+Down`	Scrolls output area down
`Ctrl+Up`	Scrolls output area up
`Ctrl+PageDown`	Scrolls output area down one page
`Ctrl+PageUp`	Scrolls output area up one page
`Ctrl+End` `Shift+Ctrl+PageDown`	Scrolls output area to bottom
`Ctrl+Home` `Shift+Ctrl+PageUp`	Scrolls output area to top
`Ctrl+S`	Splits command block at cursor (only available if syntax highlighting on)

Default Keybindings for Forms

The following default bindings are used for forms:

Key Sequence	Action
`Alt + O`	Click the *OK* button
`Alt + A`	Click the *Apply* button
`Alt + D`	Click the *Defaults* button
`Alt + H`	Click the *Help* or *Hide* button
`Alt + C`	Click the *Cancel* or *Close* button
`Alt + M`	Click the *More* button
`Alt + Y`	Click the *Yes* button
`Alt + N`	Click the *No* button
`Alt + L`	Click the *Last* button (Available in option forms)
`Alt + Q`	Click the *Quit* button (Available in option forms)

Rules for Bindkey Use with Keypads

When using bindkeys with keypads, the following rules should be considered:

Note: If you display Virtuoso remotely, for example, using a VNC viewer or Hummingbird, there can be issues with the recognition of the `NumLock` state.

1. If `NumLock` is *on*, only **numeric bindings**; `KP_0` through `KP_9`, `KP_Decimal`/ `KP_Delete` and the surrounding keys (`KP_Divide`, `KP_Multiply`, `KP_Enter`, and so on) will invoke bindkeys.

   For example, if `NumLock` is *on*, and `KP_Home` or `Home` is bound, but not `KP_7`, then pressing the `7` key on the keypad will fail to invoke anything. Rather, it would only invoke a bindkey for `KP_7`.

2. If `NumLock` is *off*, then **non-numeric keypad** binding will be searched for first, followed by the numeric keypad binding, then finally the non-keypad binding.

   For example, if `Numlock` is *off*, and `Home` is bound, but not `KP_7` or `KP_Home`, then pressing the `7` key on the keypad will invoke the bindkey for `Home`. If `Home` and `KP_7` are both bound, but not `KP_Home`, then the bindkey for `KP_7` will be invoked. And, finally, if `KP_Home` is bound, `KP_Home` will be invoked.

   Additionally, if `NumLock` is *off*, then the keys that surround the keypad (for example, `KP_Divide`, `KP_Multiply`, and `KP_Enter`) will work the same as if `NumLock` were *on*.

Note: The symbol keys do not match the printable characters ("/", "*", and so on), however the `Enter` key on the keypad will match `"Enter"` if `"KP_Enter"` is not defined, whether or not `NumLock` is *on*. Also, for rule 2 above, `KP_Decimal` is considered to be a "numeric" binding as it shares a key with `KP_Delete`. Therefore, if `NumLock` is *off*, then pressing that key will match "`KP_Delete`" if that is defined, then "`KP_Decimal`", then finally "`Delete`". If `NumLock` is *on*, it will only match "`KP_Decimal`".

Access Keys

Menu access keys provide keyboard access to functionality and application menus without the need to use mouse selections.

For example, selecting the `Alt + F` access keys together will display the contents of the *File* banner menu (for example in the Virtuoso Schematic Editor XL).

Menu access keys should not be confused with bindkeys, as bindkeys do not require a menu to be displayed before a bindkey “shortcut” can be used.

For example if you want to open the Editor Options form (for example in the Virtuoso Schematic Editor) directly, you can use the “O” (`Shift + O`) bindkey. It is not necessary to use the *Edit* menu access keys (`Alt + E`) to first of all display the contents of the *Edit* menu.

Note: When you have displayed a menu’s content further menu access keys may be used to run any of the sub-menu options listed.

Important

When a menu’s content is on display you cannot use bindkeys.

Once you have used access keys to **display** the contents of the *Launch* menu you can then use a further access key to select an application menu to be **added** to the banner menu. **After** an application menu has been **added** to the banner menu you can then use their associated access keys to **display** their menu content.

For example, to display the content of the *AMS* menu on the banner menu you would:

1. Select the access keys `Alt + L` (this will **display** the content of the *Launch* menu).
2. Select the access key `M` (this will **display** the contents of the *Mixed Signal Options* sub-menu, where the *AMS* option can be found).
3. Select the access key `A` (this will **add** the *AMS* menu to the banner menu)
4. Select the access keys `Alt + A` (this will **display** the content of the *AMS* menu on the banner menu).

Note: For information on application specific access keys see the relevant application user guides, for example the Quick Reference - Menu Access Keys section in the *Virtuoso Schematic Editor User Guide*.

B

Form Descriptions

See

- Add Bookmark
- Auto Checkin Preferences Form on page 350
- Auto Checkout Preferences Form on page 351
- Cdsenv Editor Form on page 352
- Check In Form on page 354
- Check Out Form on page 355
- Choose Default Workspace on page 356
- Choose Workspace to Load on page 357
- Close and Purge Data Form on page 358
- Close Opened Cellviews Form on page 359
- New File Form on page 368
- Edit Bookmark Properties on page 360
- Edit Library Path Form on page 361
- Exit Dialog Box on page 362
- File Preferences Form on page 363
- Bindkey Configuration Form on page 353
- Make Read Only Form on page 364
- New Hierarchy Form on page 365
- New Library Form (from the CIW) on page 366
- Open File Form on page 369

Add Bookmark

Name displays the name of the current design cellview or cellviews. You can type a different name in this field to personalize the name of your bookmark.

Description lets you type a description for your bookmark. This text appears in the *Description* column of the Bookmarks Manager window.

Add to Bookmarks Toolbar lets you add the bookmark to the *Bookmarks* toolbar.

Bookmark All Tabs lets you bookmark all open tabs in your session window as a composite bookmark.

OK performs the add action you specified.

Cancel aborts the add action and closes the form.

Help opens help information in Cadence Help.

Auto Checkin Preferences Form

Auto Checkin Preferences lets you select whether or not you want to be prompted when you check in properties and files or cellviews if you have selected the automatic checkin feature.

When auto checking in properties and files:

always ask me opens a checkin prompt form whenever you attempt to close and save a file that is not a cellview, such as a property file or data file, in a managed library.

never ask me does not open a checkin prompt form whenever you attempt to close and save a file that is not a cellview, such as a property file or data file, in a managed library.

When auto checking in cellViews:

never ask me does not open a checkin prompt form whenever you attempt to close and save a cellview in a managed library.

always ask me opens a checkin prompt form whenever you attempt to close and save a cellview in a managed library.

never auto checkin does not check in the cellview whenever you attempt to save and close a cellview in a managed library.

always auto checkin checks in the cellview whenever you save and close a cellview in a managed library.

Auto Checkout Preferences Form

Auto Checkout Preferences lets you select whether or not you want to be prompted when you check out properties and files or cellviews if you have selected the automatic checkout feature.

When auto checking out properties and files:

always ask me opens a checkout prompt form whenever you attempt to open a file that is not a cellview, such as a property file or data file, in a managed library.

never ask me does not open a checkout prompt form whenever you attempt to open a file that is not a cellview, such as a property file or data file, in a managed library.

When auto checking out cellViews:

always ask me opens a checkout prompt form whenever you attempt to open a cellview in a managed library.

never ask me does not open a checkout prompt form whenever you attempt to open a cellview in a managed library.

Cdsenv Editor Form

SeeUsing the Cdsenv Editor.

Bindkey Configuration Form

See Configuring Application Bindkeys.

Check In Form

Files to Check In displays the files you selected in the Library Manager for checkin. You can prevent files shown in the *Files to Check In* fields from being checked in by clicking on the button to the left of each cell view name.

Description (Optional) lets you enter information about the cellviews checked in.

Check Out Form

Files to Check Out displays the files you selected in the Library Manager for checkout. You can prevent files shown in the *File to Check Out* fields from being checked out by clicking on the button to the left of each cell view name.

Choose Default Workspace

Workspace is a drop-down combo box in which you can select the workspace you want to use as the default.

OK sets the selected workspace as the default.

Cancel aborts the choose action and closes the form.

Help opens help information in Cadence Help.

Choose Workspace to Load

Workspace is a drop-down combo box from which you can select a workspace to load and apply to the current view type.

OK causes the program to configure your session window using the workspace you selected.

Cancel aborts the load action and closes the form.

Help opens help information in Cadence Help.

Close and Purge Data Form

All selects all the cellviews in all the listed libraries.

None deselects any selected cellviews.

For all modified data sets how Virtuoso handles modified data on *File – Close Data* selection:

prompt to always save data on close.

save data automatically on close.

discard unsaved data if Virtuoso closed using *File – Close Data*.

Library displays the names of the libraries containing the cellviews that you had opened. You can close and delete their data from virtual memory.

Cell/Library File displays the names of all the cells or files corresponding to the library shown in the library column that you had opened either for viewing or editing. You can close and delete their data from virtual memory.

View/Cell File displays the name of the cellview, for example, `schematic`, `symbol`, and so on that you have opened for viewing or editing. You can close and delete their cellview or file data from virtual memory.

Mode displays whether the opened cellview is in read-only mode (which means you cannot edit it) or in edit-only mode (which means you can edit it).

Close Opened Cellviews Form

The list box displays the open cellviews and allows you to save and close them.

Library lists the name of the library whose cellviews you have opened either for viewing or editing.

Cell displays the name of all the cells corresponding to the library that you have opened either for viewing or editing.

View displays the names of the cellviews that you have opened.

Status provides you with the status of these opened cellviews—if the cellviews have not changed, this row is blank; if the cellviews have have been changed, `modified` appears.

Save modified cellviews allows you to save the opened cellviews before defragmentation. All modified cellviews are saved by default.

Edit Bookmark Properties

Name displays the name of the current design cellview or cellviews. You can type a different name in this field.

Description lets you type a description for your bookmark. This text appears in the *Description* column of the Bookmarks Manager window.

Add to Bookmarks Toolbar lets you add the bookmark to the *Bookmarks* toolbar.

OK performs the edit action.

Cancel aborts the edit action and closes the form.

Help opens help information in Cadence Help.

Edit Library Path Form

Library lets you choose a library name and path to edit.

Name lets you give to the library.

Directory lets you select the directory to which you assign the library if you are not using design management. If you are using design management, you must put the library in the design-managed workarea from which you started the software.

Design Manager lets you choose your design management setup.

Cadence DM lets you open your library under design management.

No DM provides no design management features such as checkin and checkout for your library.

Exit Dialog Box

Yes ends this design session and closes all windows.

No closes the dialog box but does not exit the software.

Help opens help information in Cadence Help.

File Preferences Form

File Preference lets you specify the maximum number of entries that can appear on the recently used file list.

Open Browser Automatically For lets you decide which forms, if any, you want the Library Browser to open up with.

Open Form

yes opens the Library Browser automatically, along with the Open File form, each time you choose *File – Open* from the CIW.

no does not open the Library Browser automatically, along with the Open File form, each time you choose *File – Open* from the CIW.

Create Instance Forms

yes opens the Library Browser automatically each time you choose *Add – Instance* or *Create – Instance* from a schematic or layout window.

no does not open the Library Browser automatically each time you choose *Add – Instance* or *Create – Instance* from a schematic or layout window.

Other Forms

yes opens the Library Browser automatically each time you select other commands you set in your `.cshrc` file.

no does not open the Library Browser automatically each time you select other commands.

CIW Preferences

Prompt On Exit

yes brings up the Exit dialog box automatically each time you choose *File – Exit* from the CIW.

no does not bring up the Exit dialog box automatically each time you choose *File – Exit* from the CIW.

Make Read Only Form

Select one or more cellViews to make read-only. You will be prompted to save any modified data.

All allows you to make all the cellviews you have opened readonly. If you have changed the data in any cellview, you get a warning asking if you need to save the modified data.

None does not change any of the selected cellviews to readonly.

Library provides the name of the library whose cellviews you have opened for either viewing or editing.

Cell displays the name of all the cells corresponding to the library that you have opened for either viewing or editing.

View displays the names of the cellviews that you have opened.

Window displays the window number containing the cellview you had opened for either viewing or editing.

Modified indicates whether a cellview has been modified.

New Hierarchy Form

Template

Built-in shows available templates for view lists, stop lists, and constants that contain more detailed hierarchy bindings. The default is *Other*; it allows you to specify a new template.

Name specifies the name you give to a new view list and stop list template. You enter the view names you want for this new template in the *Switch List* and *Stop List* fields.

Top Cell

Library specifies the name of the library where the cell at the highest level (the root cell) of the configuration resides.

Cell specifies the cell at the highest level of the configuration (the root cell).

View specifies the name of the configuration file.

Browse opens the Library Browser.

Global Bindings

Library List specifies, for designs not in Design Environment, libraries for cells that do not have library bindings. The libraries are listed in the order in which they are to be searched.

View List specifies the cellviews you want in your configuration in order of selection. It applies to every level of the configuration. The view list determines which view is selected for every object in the design, unless overridden by a cell or instance binding.

Stop List specifies cellviews that are at the leaf level; that is, they do not have any other levels of hierarchy below them and are not to be expanded further.

Description lets you enter a brief description (purpose, structure, etc.) of the configuration.

New Library Form (from the CIW)

See also: New Library Form (from the Library Manager).

Library lets you choose your new library name and path.

Name assigns a name to the new library.

Directory (non-library directories) lets you select the directory into which you put the new library.

Compression Enabled check box writes OpenAccess data to library in a compressed format.

Technology File lets you choose your technology file setup.

Compile an ASCII technology file assigns a copy of the sample technology file (shipped with the software) to the new library for you to edit.

Reference existing technology libraries lets you reference existing technology libraries.

Attach to an existing technology library opens a form that lets you select a technology file to attach to an existing library in your library definitions file.

Do not need process information creates the library without an attached technology file.

Design Manager lets you choose your design management setup. If more than one design management option is available, a cyclic field is displayed with your choices, otherwise a single option will be shown. When no design management environment is available, *No DM* will be shown. In this scenario, your library will open with only basic design management features available.

Important

Alternative New Library forms will be displayed dependent upon whether the form was accessed from the CIW or directly from the *Cadence Library Manager*. For more information see Creating a Library in Library Manager in the *Cadence Library Manager User Guide*).

New Library Form (from the Library Manager)

See also: New Library Form (from the CIW).

Library lets you choose your new library name and path.

Name assigns a name to the new library.

Directory lets you select the directory into which you put the new library.

Design Manager lets you choose your design management setup.

Use <design management system> lets you select the design management system to be used. *Use NONE* will be shown if no design management system is available. This option will also be grayed out in this scenario.

Use No DM opens your library with no specific design management features for your library. This option will be grayed out as a choice unless there is an alternative to use a design management system.

Compression Enabled check box writes OpenAccess data to library in a compressed format.

Important

Alternative New Library forms will be displayed dependent upon whether the form was accessed from the CIW or directly from the *Cadence Library Manager*. For more information see Creating a Library in Library Manager in the *Cadence Library Manager User Guide*).

New File Form

Library lets you select the name of the library where you want to put the new view. The drop-down combo box shows only libraries specified in your library definitions file.

Cell lets you type the name of the cell to create.

View lets you type the name of the view to create.

> **Note:** For the *Cell* and *View* name fields, only legal identifiers in the CDBA name space can be used. For example, *white spaces* cannot be used in a field name.

Type lets you select from the drop-down combo box what kind of view you want to create, for example *layout*, *schematics*, *adexl*, and so on.

Open with lets you choose the application that you want to associate the new view with, for example *Schematics L*, *ADE Explorer*, and so on. The options displayed here will be dependent on the view type you have already selected.

Always use this application for this type of file select this option to always associate a particular type of view with the application that you have selected to open it with.

Library path file displays the path to your library definitions file. If the path is too long to be fully displayed, you can put the cursor in this field and use the arrow keys to display the rest of the path. You cannot edit this field.

Open File Form

Library Name lets you select the name of the library containing the cellview to open. Only libraries specified in your library definitions file appear in this drop-down combo box.

Cell Name lets you type, or select from the *Cell Names* list box, the name of the cell to open.

View Name lets you specify the name of the view to open.

Browse opens the Library Browser.

Mode lets you choose how you want to use the opened cellview.

edit displays the cellview and permits editing.

read displays the cellview.

Library path file displays the path to your library definitions file. If the whole path is too long to be displayed, you can put the cursor in this field and use the arrow keys to display the rest of the path. You cannot edit this field.

Cell Names displays the names of all the cells in the library shown in the *Library Name* drop-down combo box.

Open in lets you choose how you want to open the cellview.

new tab opens the cellview on a new tab in your current session window.

current tab opens the cellview on the current tab in your current session window.

new window opens the cellview in a new session window.

OK opens the cellview according to the choices you specified.

Cancel closes the form without doing anything.

Defaults restores default settings on this form.

Help opens help information in Cadence Help.

Print Form

Print Command is the UNIX print command plus the name of the file to which you saved the text contents (using the Save As form).

OK submits the specified print command.

Cancel cancels the save operation and closes the form.

Help opens help information in Cadence Help.

Save .cdsenv file Form

SeeCustomizing How Updated Environment Variables are Saved.

Save As Form

Look in is the directory to which you want to save the text file.

File Name is the new name you want to assign to the file. The file is saved in your current directory unless you specify a different directory. If a file already exists under that name, the currently displayed data is not saved: you must type another file name.

File type is a drop-down combo box of available file types for filtering what you see in the list area of the form. The default selection is *All Files (*)*.

Save saves the specified file and closes the form.

Cancel cancels the save operation and closes the form.

See also Print form.

Save Cellviews Form

Save these cellViews before closing? lists the library name, cell name, and view name of each cellview to which you have made unsaved changes.

When the button to the right of the view name is selected, changes to that cellview are saved when you click *OK*.

When the button to the right of the view name is not selected, changes to that cellview are not saved when you click *OK*.

All indicates that all of the changed cellviews will be saved when you click *OK*.

None indicates that none of the changed cellviews will be saved when you click *OK*.

Save Defaults Form

Tools To Save lets you specify the applications, such as schematic, for which you want to save environment variables.

All possible tools specifies all applications in the *Available Tools* list box.

All loaded tools (changed values only) specifies all applications shown in the *Currently Loaded* list box. This option saves only the values that are different from default values.

Loaded tools by name (changed values only) specifies the applications you type in the *Tool Names* field. When you choose this option, the *Tool Names* field becomes editable. This option saves only the values that are different from default values.

Variables To Save lets you specify the criteria to use when saving environment variables. These options are available when you select the *All loaded tools* or *Loaded tools by name* options from *Tools to Save*.

Modified variables only specifies that only variables that have been modified in the session will be written out.

Modified and different from defualt specifies that variables that have been modified with a value different from the default value will be written out.

All tool variables specifies all variables will be written out, whether modified or not.

Available Tools shows the applications for which you have a license.

Currently Loaded shows the applications for which environment variables are loaded.

Tool Names lets you type names of individual applications for which you want to save the default values. The default is `ALL`.

Save To File is the name of the file in which you want to save the information. The default file name is `~/.cdsenv`. You can save and use multiple environment files to suit the needs of your application.

File Status lets you decide how you want to save your changes.

Overwrite overwrites your `.cdsenv` file.

Merge values saves the values into your `.cdsenv` file. It does not delete pre-existing unmodified values.

Retain values saves the values to a new file. You must type a different file name in the *Save To File* field.

Save Modified Data

Select one or more objects to save from virtual memory lets you select or deselect all open and modified cellviews and data in the form.

All selects all the cellviews in all the listed libraries.

None deselects any selected cellviews.

Library displays the names of the libraries containing the modified cellviews.

Cell/Library File displays the names of all cells or files corresponding to the library shown in the library column.

View/Cell File displays the name of the modified view. For example, `schematic`, `symbol`, and so on.

View Type displays the real view type of a cellview. In a cellview, view name can be user defined, for example, view name can be `layout1`. However, view type is always same for a specific type of data, such as `maskLayout`, `schematic`, and so on.

Mode displays whether the cellview is in `edit` mode. Only data in `edit` is listed here.

Status displays the status of data in specific window. The status is always `modified`, however, if a cellview is open in a window, the window number is included, for example, `win4:modified`, `win5:modifed` and so on.

Related links

Saving Modified Data

ddsHiSaveData

Save Session Form

File Name is the name of the file in which you want to save the information. The default file name is `cdsSession.save`. You can rename this file.

Save View

Name lets you type a name for the view.

OK performs the save action and closes the form.

Cancel aborts the save action and closes the form.

Apply performs the save action and leaves the form open.

Help opens help information in Cadence Help.

Set Log File Display Filter Form

Show: Marked check boxes indicate those items that will appear in the output area of the Command Interpreter Window (CIW)

- Return values from typed-in commands
- Error message output
- Warning message output
- Standard message output
- Accelerated input (menu and bindkey commands, form and field value settings and callbacks, etc.)
- Return values from accelerated input
- Prompt output

Show File Form

File Name is the name of the file to view. If you do not specify a path, the SKILL search path is used.

Software Product License Management Form

Checked Out License*s* are licenses currently running on your desktop. This pane updates when licenses are checked out by the applications or are started in the *Task Assistant* menu in the editing environments.

Relevant Licenses are licenses available with the platform that is running.

Token Information provides details of the various ADE GXL and VLS EXL capabilities; including information on the features in use, the current availability of tokens, and the number of tokens required to run each capability.

Users shows token license usage and availability of the license you highlight. Update this form by choosing the license in either pane and clicking the *Users* button. `Control space` deselects highlighted licenses.

Text Editor Form

File Name is the name of the text file to edit. If no path is specified, the working directory is searched.

Unable to check out Form

When all available licenses have been checked out, this form lists licenses currently in use and who is using them.

Product identifies the product number that cannot be checked out.

#Licenses specifies how many licenses are available for the application.

User idenitifies current users who have a license checked out for the application.

Host specifies the host system where the application resides.

Product Description provides the name or short description of the application.

User Preferences Form

Window Controls let you set window appearance automatically.

Place Manually controls the window placement.

When turned off, places graphics windows and text windows by default in the horizontal center of the screen, slightly above the vertical center, in a size and shape determined by the Cadence software.

When turned on, lets you determine the position, size, and shape of windows. You place a window by clicking to place one corner of the window, dragging the cursor to the opposite corner, and releasing the button when the window is the size and shape you want. The command does not affect text editor windows.

Create New Window When Descending controls the type of view displayed when you go up and down the design hierarchy. Not all applications make use of this feature.

When turned off, continues to display the original view type.

When turned on, displays another window with the new view type.

Scroll Bars controls visibility of scroll bars.

When turned off, creates new graphics windows without scroll bars.

When turned on, creates new graphics windows with scroll bars on the right and bottom edges. Existing windows are not affected.

Tear-Off Menus controls whether future menus have tear-off handles.

When turned off, future menus have tear-off handles.

When turned on, future menus do not have tear-off handles.

Menu Shortcuts controls whether menu shortcuts are available.

When checked, you can type *Alt* together with the underlined character for a menu (such as the *F* for the *File* menu in the CIW) to access that menu, and you can type the underlined character on a menu item (such as the *O* in *Open* on the *File* menu) to access that item.

When not checked, you cannot access menus or menu items using keyboard shortcuts.

Mouse Prompts controls whether and where mouse prompts appear.

Top if you want mouse prompts to appear along the top edge of each new window.

Bottom if you want mouse prompts to appear along the bottom edge of each new window.

None if you do not want the mouse prompts to appear at all.

Command Controls affect mouse and options forms behavior.

Infix (No Click is necessary for first point) controls mouse behavior in design windows.

When turned off, prompts for the starting point of a line or shape.

When turned on, uses the point at which you use a pop-up menu as the first data point in any command you choose from the pop-up menu.

Note: *Infix* and *Options Displayed When Commands Start* should not be used together in a GNOME window manager environment unless the mouse focus setting is set to be "*focus under mouse*" or "*focus strictly under mouse*".

Options Displayed When Commands Start controls the behavior of options forms.

When turned off, does not display an options form for changing command settings. Press `F3` to see the options form.

When turned on, displays an options form when you select a command or reach a stage in a command where you might need to change the settings.

Undo Limit specifies how many commands you can undo with the *Undo* command. *Undo* undoes the most recent commands in reverse of the order in which they were used. You can select a limit of 0 or 128. A value of 0 disables undo.

Nest Limit specifies how many levels of commands can be nested. You can select a limit from 1 to 20 commands using the up and down arrows in the spin box.

Double Click Time sets the interval (in milliseconds) in which two mouse clicks are interpreted as a double-click. You can select intervals from 200 to 1000 milliseconds by dragging the slider.

Beep Volume sets the volume of the beep used in the Cadence software. You can select intervals from -100 to +100 by dragging the slider.

Web Browser specifies a browser executable. The directory where the executable is must be in your path. The default is `firefox`.

CIW Controls affect the appearance and configuration of the Command Interpreter Window.

Output History specifies the maximum number of commands contained in the output log. If you specify a number that is smaller than the current setting, the program truncates the contents of the output log. The minimum valid value for this setting is 100.

Input Buffer Lines specifies the number of previous commands that appear in the input area.

Input Area Lines specifies the initial number of lines (size) of the input area, limited to the available space in the window. If you specify more lines than are available in the window, the program makes additional space available to allocate to the input area by hiding the output history area.

Enter Key Executes Command specifies whether or not you can press *Enter* to execute the command typed on the input line.

When turned on, you can press *Enter* to execute the command typed on the input line.

When turned off, you must place the cursor at the end of the input line and press *Enter* to execute the command.

Raise CIW on: Specifies whether the CIW should be raised to the front when a *Warning* and/or an *Error* has been displayed in the log window. The default is to not display the CIW in either of these situations.

The following equivalent environment variables can also be used:

```
ui raiseCIWonError boolean t/nil
ui raiseCIWonWarning boolean t/nil
```

Dashboard Controls affects the display of license activity indicators.

Display Dashboard Indicators specifies whether or not to display dashboard indicator information, related to resource license activity.

License Activity specifies where dashboard controls are displayed, either in the *CIW*, the *CIW+Session* windows, or that they are *Hidden*.

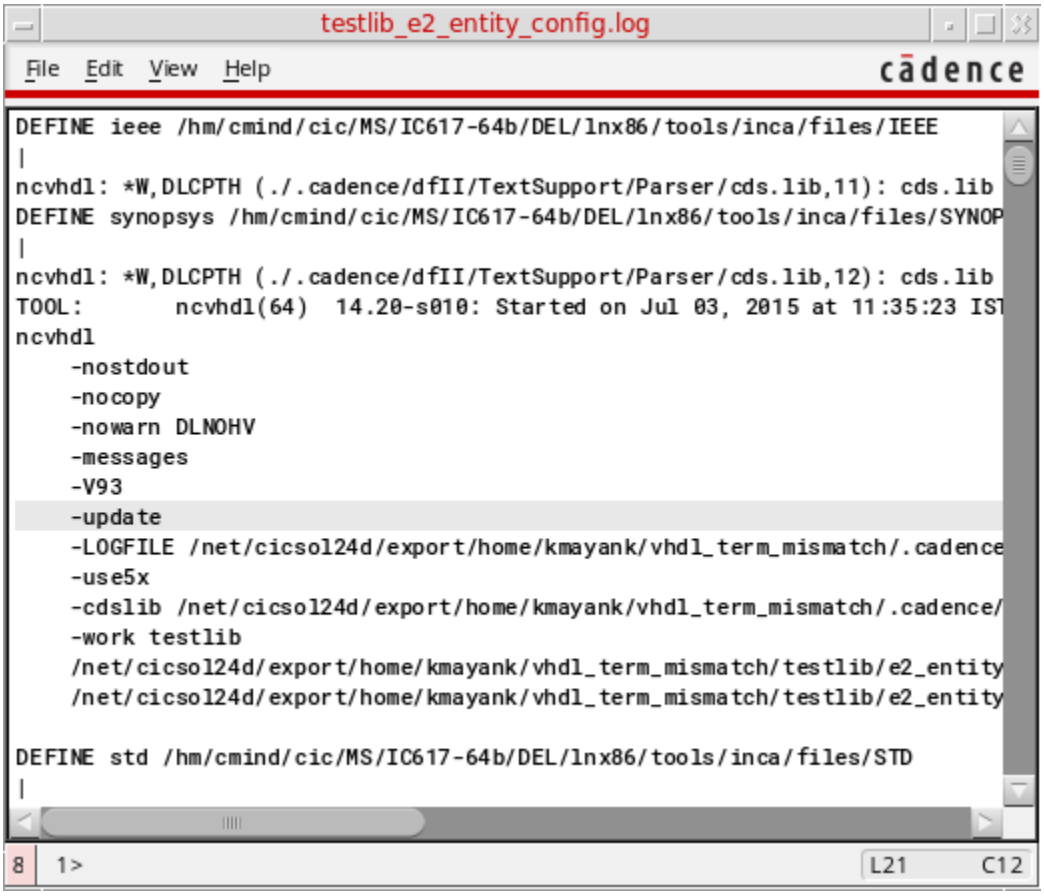
ViewFile Window

Lets you view and manage its contents but does not let you edit them.

A viewfile window is used in various Virtuoso applications. For example, layout uses a viewfile window to display the design hierarchy tree, bindkey editor displays the bindkey commands in a viewfile window, TechDB Checker uses the viewfile window to display its log file and reports, and tools, such as Assura, ADE, VHDL toolbox and NC Verilog integration window use a viewfile window as a netlist viewer.

Note: For more information about the content of a viewfile window that is used in a particular application, check the respective application document.

A typical viewfile window is as shown below:



The following subsections describe the command menus on a viewfile window:

- File Menu

- Edit Menu
- View Menu
- Help Menu

File Menu

Menu	Description
Open	Opens a file for viewing in the current viewfile window
Save	Saves the information in the viewfile window to the log file
Save As	Saves the information in the viewfile window to a new file When you start this command, the Save As form appears. You use the form to name the file where you want to save the information.
Auto Update On	Causes the simulator to update the information in the view window automatically as the simulation continues
Auto Update Off	Prevents the simulator from updating the information in the viewfile window as the simulation continues
Close Window	Closes the viewfile window

Edit Menu

Menu	Description
Copy	Copies the selected text to the clipboard. You can also click the *Copy* command before selecting the text to be copied.
Select All	Selects the entire contents of the file

Menu	Description
Find	Searches for text strings in the viewfile window When you run this command, the Search form appears. You use the form to specify the text string you want to find, and to control the search. *Search for* specifies the text string you want to find. *When Found* specifies whether to select or deselect the text string. *select* highlights the text string. *deselect* turns off highlighting of the text string. *scroll to next match* scrolls to the section of text that contains the next occurrence of the text string. If the next occurrence is already displayed within the viewing window, the window does not scroll. *Match Options l*ets you control for partial hits and case. *whole word* specifies a search for the exact string. *exact case* specifies a search for the case (upper or lower) that appears in the *Search for* field. *Wrap Around* (default) continues the search from the beginning of the file. *Scan the Whole File*, when enabled, finds and highlights all occurrences of the text string you specify.
Go to Line	Searches for the line number you specify and highlights the specified line. When you run this command, the Go to Line form appears. You use this form to search for a specific line number and to move to the specified line. *Go to line* field specifies the line number to which you want to move to.

View Menu

Menu	Description

Line Numbers	Displays the following sub-menu options:
	■ *Inline*: Shows the line numbers on the left side (sidebar) of the log file.
	You can enable sidebar by using the following variable in CIW:
	`(envSetVal "ui.text" "showLineNumbersInSideBar" 'boolean t)`
	Otherwise, to enable sidebar, you can add the following syntax in the `.cdsenv` file:
	`ui.text showLineNumbersInSideBar boolean t`
	■ *Status Bar*: Displays the line and column number in the lower-right corner of the log file.

Highlight Current Line	Highlights the current line in gray.

Help Menu

Opens the current document that describes viewfile window commands.

For more information about the viewfile window SKILL functions, see the "Viewfile Window" chapter in the *Cadence User Interface SKILL Reference Guide*.

What's New Search

Search For lets you type a search string.

Match Entire String lets you specify whether you want the program to match the entire string.

yes indicates that you want the program to match the entire search string that you typed in the *Search For* field.

no indicates that you want the program to match words within the search string.

Match Case lets you specify whether you want the program to match casing (uppercase/lowercase).

yes indicates that you want the program to match casing.

no indicates that you want the program to ignore casing when finding a match.

Wrap Search lets you specify whether the program should wrap around to the top of the search document when it comes to the end.

yes indicates that you want the program to wrap around to the top of the search document when it comes to the end as it searches.

no indicates that you want the program to stop searching when it comes to the end of the search document.

C

CDBA Environment Variables

This appendix provides information on the names, descriptions, and graphical user interface equivalents for the CDBA environment variables.

List of CDBA Environment Variables

AlternateFoundryCG

See `AlternateFoundryCG`.

Related Topics

List of CDBA Environment Variables

copyMPAttributes

See `copyMPAttributes`.

Related Topics

List of CDBA Environment Variables

dbAddCellNameToInstNamePrefix

```
cdba dbAddCellNameToInstNamePrefix boolean { t | nil }
```

Description

Appends the cell name to the existing instance name prefix in instance names. The default is `nil`.

For example, if the instance name prefix is `INST` and you set `dbAddCellNameToInstNamePrefix` to `t`, the name `INST_`*`cellname_number`* is generated, were, *`cellname`* is the cell name and *`number`* is a system generated unique number.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbAddCellNameToInstNamePrefix")
envSetVal("cdba" "dbAddCellNameToInstNamePrefix" 'boolean t)
envSetVal("cdba" "dbAddCellNameToInstNamePrefix" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

dbAllowStdViaCutLayerOverride

```
cdba dbAllowStdViaCutLayerOverride boolean { t | nil }
```

Description

Allows the `cutLayer` parameter for standard vias and standard via variants to be overridden by all SKILL and ITKDB functions. The default is `t`.

Note: For the standard via variant defined in the technology file, the specified `cutLayer` is always ignored. Instead the `viaDefs cutLayer` is used.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbAllowStdViaCutLayerOverride")
envSetVal("cdba" "dbAllowStdViaCutLayerOverride" 'boolean t)
envSetVal("cdba" "dbAllowStdViaCutLayerOverride" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

dbArrayInstNamePrefix

```
cdba dbArrayInstNamePrefix string "array_instance_prefix"
```

Description

Prepends the specified string to the names of array instances. The default is `M`.

For example, if the specified prefix is `M` and, the name `M_arrayInstanceName` is generated.

GUI Equivalent

None.

Examples

```
envGetVal("cdba" "dbArrayInstNamePrefix")
envSetVal("cdba" "dbArrayInstNamePrefix" 'string "P")
```

Related Topics

List of CDBA Environment Variables

dbEnableRouteObservers

```
cdba dbEnableRouteObservers boolean { t | nil }
```

Description

Enables observers that are used to observe the route author updates. The default is `nil`, which means that observers are disabled.

For more information, see the `dbGetRouteAuthor` SKILL function.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbEnableRouteObservers")
envSetVal("cdba" "dbEnableRouteObservers" 'boolean t)
envSetVal("cdba" "dbEnableRouteObservers" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

dbFigGroupNamePrefix

```
cdba dbFigGroupNamePrefix string "figure_group_name_prefix"
```

Description

Prepends the specified string to the name of a figure group. The default is `group`.

For example, if the specified prefix is `group` and, the name `group_figGroupName` is generated.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbFigGroupNamePrefix")
envSetVal("cdba" "dbFigGroupNamePrefix" 'string "mgroup")
```

Related Topics

List of CDBA Environment Variables

dbInstNamePrefix

```
cdba dbInstNamePrefix string "custom_instance_name_prefix"
```

Description

Prepends the specified prefix to the name of an instance. The default is `I`.

For example, if the specified prefix is `I`, the name `I_instName` is generated.

GUI Equivalent

None.

Examples

```
envGetVal("cdba" "dbInstNamePrefix")
envSetVal("cdba" "dbInstNamePrefix" 'string "INST")
```

Related Topics
List of CDBA Environment Variables

dbLogPcellWarnings

```
cdba dbLogPcellWarnings boolean { t | nil }
```

Description

Redirects warning messages generated during Pcell evaluation to the Pcell error log file.

The default is `nil`, which means that only the error messages are redirected to the Pcell error log file. When set to `t`, warning messages are also redirected to the Pcell error log file.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbAllowStdViaCutLayerOverride")
envSetVal("cdba" "dbAllowStdViaCutLayerOverride" 'boolean t)
envSetVal("cdba" "dbAllowStdViaCutLayerOverride" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

dbNumCPU

`cdba dbNumCPU int` *`any_positive_integer`*

Description

Specifies the number of CPUs. The default is `1`.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbNumCPU")
envSetVal("cdba" "dbNumCPU" 'int 2)
```

Related Topics

List of CDBA Environment Variables

dbUndoAcrossPurge

```
cdba.undo dbUndoAcrossPurge boolean { t | nil }
```

Description

Preserves the undo checkpoint information when purging a cellview. The default is `nil` , which means that undo checkpoint information is lost when a cellview is purged.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbUndoAcrossPurge")
envSetVal("cdba" "dbUndoAcrossPurge" 'boolean t)
envSetVal("cdba" "dbUndoAcrossPurge" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

dbUndoAcrossSave

```
cdba.undo dbUndoAcrossSave boolean { t | nil }
```

Description

Preserves the undo checkpoint from losing information while saving a cellview. The default is `nil`, which means that undo checkpoint information is lost when a cellview is saved.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbUndoAcrossSave")
envSetVal("cdba" "dbUndoAcrossSave" 'boolean t)
envSetVal("cdba" "dbUndoAcrossSave" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

defaultAttachTech

See `defaultAttachTech`.

Related Topics

List of CDBA Environment Variables

disablePartialRead

```
cdba.oa disablePartialRead boolean { t | nil }
```

Description

Disables partial read in a Virtuoso session. The default is `nil`, which means that the partial reads are allowed.

This environment variable affects the behavior of OpenAccess for the entire Virtuoso session. Its value is read by Virtuoso at startup and it does not affect the behavior of OpenAccess in any other DFII executables, such as XStream or XOasis. Any changes to the value of `disablePartialRead` during a Virtuoso session are ignored.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "disablePartialRead")
envSetVal("cdba" "disablePartialRead" 'boolean t)
envSetVal("cdba" "disablePartialRead" 'boolean nil)
```

Related Topics

List of CDBA Environment Variables

noDetailedRowCol

```
cdba noDetailedRowCol boolean { t | nil }
```

Description

Determines the syntax of mosaic tile objects in the value returned by the dbGetTrueOverlaps SKILL function.

The default is `nil`, which means that `dbGetTrueOverlaps` generates a list of mosaic tiles if they are found in the search area and if the *`g_doRowCol`* argument is set to `t`. The list of mosaic tiles is generated using the following syntax, where the row and column indexes for the mosaics are also printed:

```
; case #1 flat design
(db:MOSAIC_ID
    ((db:MOSAIC_ID 0 0)
     (db:MOSAIC_ID 0 1)
     (db:MOSAIC_ID 1 0)
     (db:MOSAIC_ID 1 1))
```

If mosaics are nested and multiple levels are queried (`startLevel != stopLevel`), the list syntax for the stop level when `noDetailedRowCol` is set to `nil` is:

```
; case #2 nested mosaics
(db:PARENT_MOSAIC_ID
    ((db:PARENT_MOSAIC_ID 0 0) db:CHILD_MOSAIC_ID)
    (db:CHILD_MOSAIC_ID 0 0)
    (db:CHILD_MOSAIC_ID 0 1)
    (db:CHILD_MOSAIC_ID 1 0)
    (db:CHILD_MOSAIC_ID 1 1)
    ((db:PARENT_MOSAIC_ID 0 1) db:CHILD_MOSAIC_ID)
    (db:CHILD_MOSAIC_ID 0 0)
    (db:CHILD_MOSAIC_ID 0 1)
    (db:CHILD_MOSAIC_ID 1 0)
    (db:CHILD_MOSAIC_ID 1 1)
    ((db:PARENT_MOSAIC_ID 1 0) db:CHILD_MOSAIC_ID)
    (db:CHILD_MOSAIC_ID 0 0)
    (db:CHILD_MOSAIC_ID 0 1)
    (db:CHILD_MOSAIC_ID 1 0)
    (db:CHILD_MOSAIC_ID 1 1)
    ((db:PARENT_MOSAIC_ID 1 1) db:CHILD_MOSAIC_ID)
    (db:CHILD_MOSAIC_ID 0 0)
```

```
    (db:CHILD_MOSAIC_ID 0 1)
    (db:CHILD_MOSAIC_ID 1 0)
    (db:CHILD_MOSAIC_ID 1 1)
)
```

When `noDetailedRow` is set to `t`, the `dbGetTrueOverlaps` SKILL function prints the result in the legacy format that does not print the row and column index for the mosaics in a flat design (case #1):

```
(db:MOSAIC_ID)
```

For nested mosaics, (case #2), the nested mosaic tiles are not printed if `noDetailedRowCol` is set to `t`:

```
(db:PARENT_MOSAIC_ID
    ((db:PARENT_MOSAIC_ID 0 0) db:CHILD_MOSAIC_ID)
    ((db:PARENT_MOSAIC_ID 0 1) db:CHILD_MOSAIC_ID)
    ((db:PARENT_MOSAIC_ID 1 0) db:CHILD_MOSAIC_ID)
    ((db:PARENT_MOSAIC_ID 1 1) db:CHILD_MOSAIC_ID)
)
```

GUI Equivalent

None

Examples

```
envGetVal("cdba" "noDetailedRowCol")
```

The following examples show how the output of the `dbGetTrueOverlaps` SKILL function changes based on how `noDetailedRowCol` environment variable is specified.

```
envSetVal("cdba" "noDetailedRowCol" 'boolean nil)

;case #1 Flat Design
> dbGetTrueOverlaps(cv cv~>bBox t 0 t)
  (db:0x1909ce1a             ; 2x2 top mosaic
    (db:0x1909ce1a 0 0)   ; 2x2 child mosaics
    (db:0x1909ce1a 0 1)
    (db:0x1909ce1a 1 0)
    (db:0x1909ce1a 1 1)
  )
```

```
; case #2 (nested mosaics)
> dbGetTrueOverlaps(cv cv~>bBox t 1 t)
  (db:0x1909ce1a                            ; 2x2 top mosaic
    ((db:0x1909ce1a 0 0) db:0x1909cd9a)   ; 2x2 child mosaic
    (db:0x1909cd9a 0 0)                   ; 2x2 child mosaic tile
    (db:0x1909cd9a 0 1)
    (db:0x1909cd9a 1 0)
    (db:0x1909cd9a 1 1)
    ((db:0x1909ce1a 0 1) db:0x1909cd9a)   ; 2x2 child mosaic
    (db:0x1909cd9a 0 0)                   ; 2x2 child mosaic tile
    (db:0x1909cd9a 0 1)
    (db:0x1909cd9a 1 0)
    (db:0x1909cd9a 1 1)
    ((db:0x1909ce1a 1 0) db:0x1909cd9a)   ; 2x2 child mosaic
    (db:0x1909cd9a 0 0)                   ; 2x2 child mosaic tile
    (db:0x1909cd9a 0 1)
    (db:0x1909cd9a 1 0)
    (db:0x1909cd9a 1 1)
    ((db:0x1909ce1a 1 1) db:0x1909cd9a)   ; 2x2 child mosaic
    (db:0x1909cd9a 0 0)                   ; 2x2 child mosaic tile
    (db:0x1909cd9a 0 1)
    (db:0x1909cd9a 1 0)
    (db:0x1909cd9a 1 1)
  )

envSetVal("cdba" "noDetailedRowCol" 'boolean t)

; case #1 flat design
> dbGetTrueOverlaps(cv cv~>bBox t 0 t)
  (db:0x1909ce1a    ; 2x2 top mosaic
  )

; case #2 (nested mosaics)
> dbGetTrueOverlaps(cv cv~>bBox t 1 t)
  (db:0x1909ce1a                             ; 2x2 top mosaic
    ((db:0x1909ce1a 0 0) db:0x1909cd9a)   ; 2x2 child mosaics
    ((db:0x1909ce1a 0 1) db:0x1909cd9a)
    ((db:0x1909ce1a 1 0) db:0x1909cd9a)
    ((db:0x1909ce1a 1 1) db:0x1909cd9a)
  )
```

Related Topics

List of CDBA Environment Variables

noTechUpRev

See noTechUpRev.

Related Topics

List of CDBA Environment Variables

D

Diagnostics

This chapter covers the following sections related to diagnostic testing and analysis:

- Reporting Application Crashes
- Measuring Graphics Performance
- Using the Performance Diagnostic Tool

Reporting Application Crashes

In Virtuoso, a report will be generated on application failure or crash. This *crash detector* report automatically records an abridged version of crash data, including command logs and stack traces (for example, SKILL calls are included in the stack trace to assist crash investigations).

This information may subsequently assist in the debugging of critical software problems and also provide a means of tracking essential data which can be useful when attempting to identify key failure trends across specific applications (as well as Virtuoso itself).

Note: A crash report is not intended to provide a description of all possible debugging techniques, nor does it provide a description of how to categorize trends or crash information.

When an incorrect termination of an application occurs, a Fatal Application Error dialog is displayed alerting you of this fact. The data is collected and made available in report format, in an ASCII text file, that includes a range of basic environment information. From the Fatal Application Error dialog you can chose to view the crash report that has been generated (*View Report*) or to *Close* the dialog.

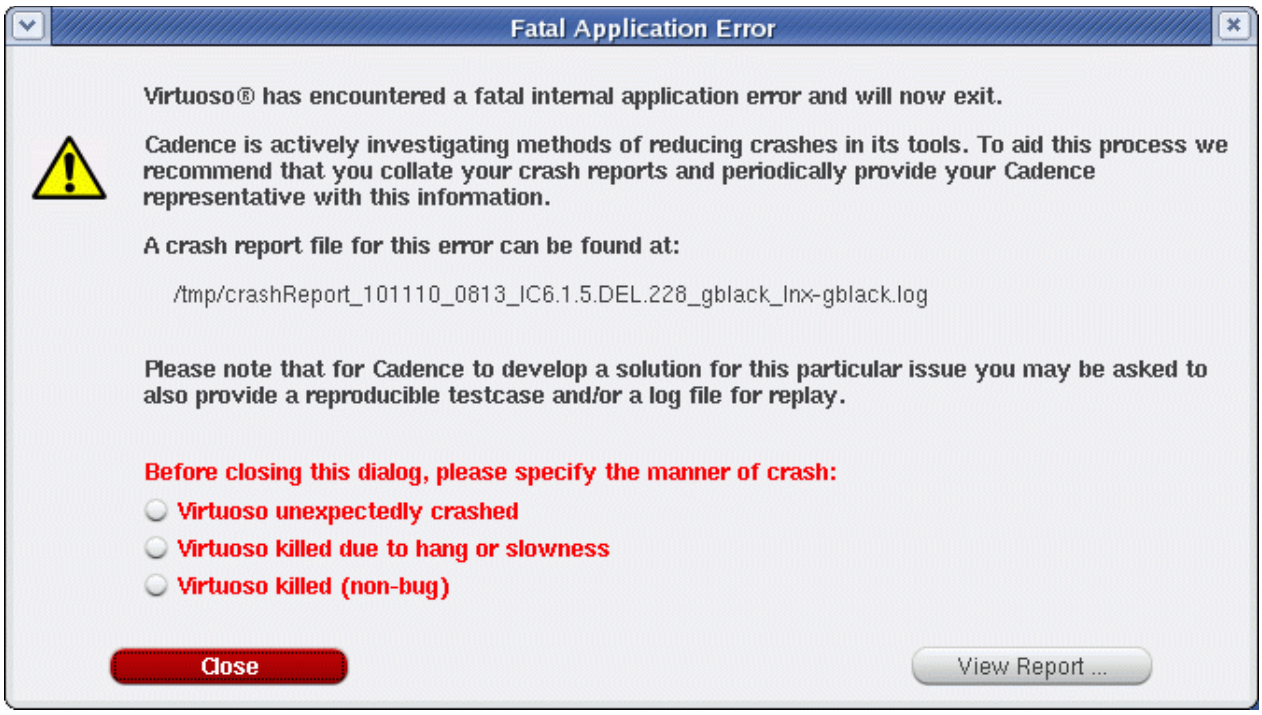


Figure D-1 A Fatal Application Error Has Occurred

Note:

- Before closing the Fatal Application Error dialog, specify the nature of the Virtuoso crash, as this will assist Cadence in its software crash investigations. In addition to the Crash Report, the option you select is also stored in `STARTINFO_FILE`.
  - Select *Virtuoso unexpectedly crashed* if the software terminated unexpectedly without manual intervention.

    If this option is selected, the following will be added to the Crash Report window:

    ```
    signal:Abort(6) ****PROCESS CRASHED****
    ```
  - Select *Virtuoso killed due to hang or slowness* if you manually terminated the software due to poor responsiveness.

    If this option is selected, the following will be added to the Crash Report window:

    ```
    signal:Abort(6) ****PROCESS KILLED DUE TO SLOWNESS****
    ```
  - Select *Virtuoso killed (non-bug)* if you manually terminated the software for a reason other than poor responsiveness.

    If this option is selected, the following will be added to the Crash Report window:

```
signal:Abort(6) ****PROCESS KILLED BY USER****
```

- If you do not select a crash reason option before selecting the Close or View Report buttons, the following will be added to the Crash Report window:

  ```
  signal:Abort(6)
  ```

- If a `SEGFAULT` happens (again) inside the `virtuoso` crash hooks, the signal will be delivered directly to, and processed by, the crash report.
- If `virtuoso` is being run in `nograph` mode, the Fatal Application Error window will not be displayed. Rather, crash report information will be sent to the `stderr` log.
- It is requested that you check the *Application was killed by user...* option if that was the case, as establishing if `virtuoso` was killed intentionally will focus attention on genuine software crashes.

If you chose *View Report*, the Crash Report window is displayed:

Figure D-2 A Generated Crash Report

Note:

- From here, you can choose to save the contents of the report to a text file (*File – Save As*) for re-use, or select *File – Close* to exit the report after viewing.
- Additionally, within the *File* menu, you can choose to *Copy* report text, *Select All* report text, or *Find* particular report information using the Find text form.
- You can also use the *View* menu to choose what categories of crash information that you want to display in the crash report. For example, you can choose to show or hide information on *Stacktrace*, *Debug*, and/or *cdsinit*.
- The crash report information is also appended to the `CDS.log` file. You can, for example, then refer to this log file for details on where the report file can be located.

  Note: On crash, a copy of the `CDS.log` file will be saved to the crash report directory, and renamed `<crashReportName>/CDS.log`. See also `SAVELOGS=ONCRASH` in the Customizing Crash Reports section. The `CDS.log` file will additionally save the date and time of crash. Also, in case Virtuoso crashed because of an external signal, the signal name will also be saved in the CDS.log file. In the earlier releases, the date and time of crash and the signal name were only saved in the crash reports.
- Relatedly, the crash report also lists the last ten errors, warnings, and commands listed in the `CDS.log` file. These crash report lines (that is, those referring to `cmd:`, `warning:`, and `error:` lines) are listed with their relevant `CDS.log` line number to aid log investigations.

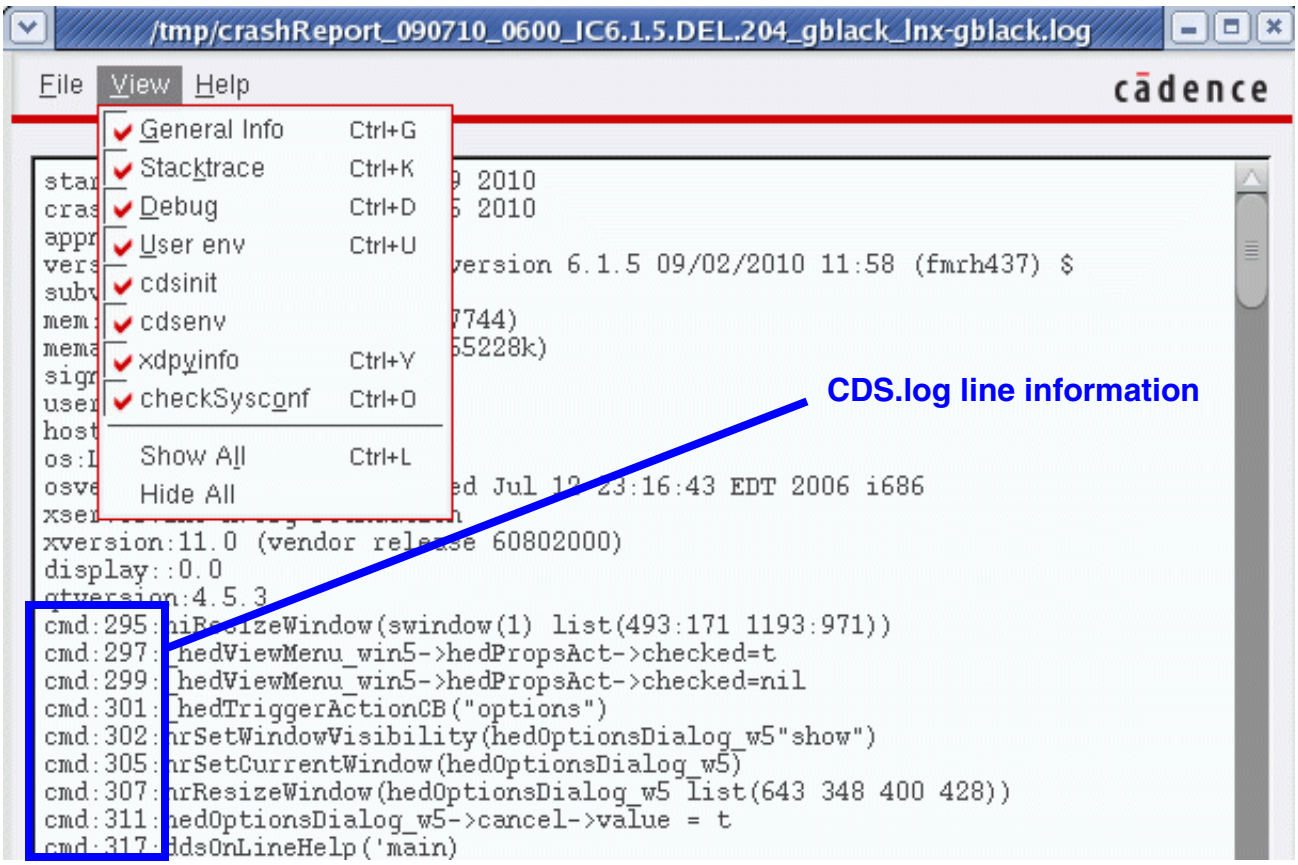


Figure D-3 Selecting Report Information to View

Customizing Crash Reports

The following name-value pairs can be set for the `CDS_ERRORLOG` environment variable to customize crash report behavior.

You only need to specify a value if you want to override the default.

Note: These environment variables are discussed in more detail in the remainder of this appendix.

Name-Value Pair (plus example)	Description
`EXTENDED=TRUE`	If set to `TRUE/YES`, additional system information will be recorded for analysis (including the values for of `.cdsenv` variables, for all loaded tools, that differ from their default values). The default setting is `TRUE/YES`. See Reporting Additional Crash Data.
`ALWAYS=NO`	If set to `TRUE/YES`, then debug information will always be created and the Fatal Application Error dialog will not be required to be displayed. The default setting is `NO/FALSE`.
`STARTINFO_FILE=abcd.out`	Used to log each session start. Specify the name of the file to log each session start/end. See Crash Trend Reporting.

Name-Value Pair (plus example)	Description

SCRIPT	If set, the user's shell/script file is executed when any application crashes. You can set the variable as:
	`setenv CDS_ERRORLOG "SCRIPT=<directory path>/<file name>.sh`
	The user script file passes the following arguments:
	`-status <exit_status>`
	`-log <path_to_CDS.log_file>`
	`-t <crash_time>`
	`-dir <log_directory>`
	`-exe <path_to_application_binary>`

DIR=.	Used to set the directory that reports are to be saved in. If you have specified a directory that does not currently exist, this directory will automatically be created. If this environment variable is not set, reports will be saved to "`/tmp`" (the name generated by `cdsGetTmpDir()`) by default. See Saving Crash Report Data.

Name-Value Pair (plus example)	Description

AUTO_DIR=<path>	If this environment variable is set, crash reporter creates a directory <path> and all crash logs are stored in <path>/year_month.
	
	For example:
	
	`>setenv CDS_ERRORLOG "AUTO_DIR=autoDir"`
	`>ls -a autoDir/16_12/`
	`=> total 24`
	`drwxrwxrwx 2 user cadence10 4096 Dec 8 19:18 .`
	`drwxrwxr-x 3 user cadence10 4096 Dec 8 19:18 ..`
	`-rw-r--r-- 1 user cadence10 5485 Dec 8 19:18 crashReport_120816_191828_IC6.1.7-64b.500.8_user_msc075.cadence.com.log`
	`-rw-rw-rw- 1 user cadence10 4252 Dec 8 19:18 crashReport_120816_191828_IC6.1.7-64b.500.8_user_msc075.cadence.com.log.CDS.log`
	
	If AUTO_DIR or DIR environment variables are not set, reports will be saved to "/tmp" (the name generated by cdsGetTmpDir()) by default.

ALL_ERRORS=TRUE	If set to TRUE, crash reports are always generated.
NAME_FMT=%PID.log	Used to define the report name. See Saving Crash Report Data.
USE_DEBUGGER=YES	If set to TRUE/YES, will attempt to attach the platform debugger on application crash to get additional stack information. The default setting is TRUE/YES. If USE_DEBUGGER has been set to YES, but no debugger has been set in the path, then the crash report will inform that a debugger was not available on exit. `debug:Warning : no platform debugger found debug:Check your system configuration`

Name-Value Pair (plus example)	Description

CDS_CRASH_DBG_SCRIPT	You can specify a different platform debugger using the CDS_CRASH_DBG_SCRIPT environment variable.
	`setenv CDS_CRASH_DBG_SCRIPT "/opt/install/ bin/my_debug.sh %PID %PROGRAM"`
	For example:
	`# example of user-defined "my_debug.sh" script`
	`# this script is passed 2 arguments: path to the binary ($1)`
	`# and process PID ($2)`
	`# It is possible to use a third-part program (debugger or`
	`# any other program that can print the stack trace)`
	`/bin/sh`
	`program=$1`
	`pid=$2`
	`/opt/install/debuggers/dbx/latest/dbx -q -nx ${program} ${pid} << EOF 2>1`
	`set prompt`
	`where`
	`detach`
	`quit`
	`EOF`
	Note: In case the user debugger cannot start, the following message is added to the crash report:
	`CDS_CRASH_DBG_SCRIPT was not started`

DISABLED=NO	If set to TRUE/YES, will disable the use of crash reporting so that no crash dialog is displayed nor a crash report generated. The default setting is FALSE/NO.

Name-Value Pair (plus example)	Description
`SAVELOGS=ONCRASH`	Lets you specify that `virtuoso` calls the `saveCdsLog.pl` log collection script at any exit, or only on crash, to automatically save the current `CDS.log`. Accepted values are: `ALWAYS/ONCRASH/OFF`. The default is `ONCRASH` which will save a copy of `CDS.log` to the crash directory on exit.
`SIGTERM=TRUE`	Lets you terminate the Virtuoso application. This name-value pair allows Virtuoso to release resources and save its state, if appropriate. The default value is `FALSE`.

Name-Value Pair (plus example)	Description

`LOG_EMAIL`	In case the session crashes (that is, a non-zero exit), sends a system-generated e-mail notification to all the e-mail addresses specified for a listed application.
	
	The format of the e-mail is as follows:
	
	`To: <application1>:<address1,address2>,<application2>:<address3,address4>`
	
	`Subject: Virtuoso log info for host:<hostname> (lnx86)`
	
	The content of the e-mail is as follows:
	
	`INST_ROOT: <Installation root of the product>`
	
	`VERSION:<Product version>`
	
	`USER: <User name>`
	
	`TIME: <Timestamp of the crash>`
	
	`REPORT: <Path of the crash report>`
	
	`LOG: <Path of the CDS.log file>`
	
	`SIGNAL: <Signame# and its description>`
	
	If `$CDS_ERRORLOG LOG_EMAIL=<`*`E-Mail list`*`>` is set, then in case of application crash, the first 50 lines of `CDS.log` comes as an attachment in an e-mail.

For example:

```
setenv CDS_ERRORLOG "SAVELOGS=ONCRASH EXTENDED=TRUE ALWAYS=NO STARTINFO_FILE=/
home/genel/startTimes_`date +%b%Y` DIR=/home/genel/crashes
NAME_FMT=crashReport_%DATE_%TIME_%SUBVERSION_%USER_%HOST.log USE_DEBUGGER=YES
DISABLED=NO
LOG_EMAIL=virtuoso:nicolay@cadence.com,xyz@customer.com,viva:pqr@cad.customer2.co
m,hello@world.com"
```

Note: Any values set should not contain any spaces.

Older versions of debuggers, such as GDB, may not correctly handle code that has been built using `gcc4.4`. This can result in incomplete stack traces being generated. It is therefore recommended that `$PATH` be set to the latest version of GDB or DBX, for example.If there is no debugger in the path, then no debug data will be included in the crash report.

The crash detector debugger can also be used to check the `$DEBUGGER` variable, and, if it is defined, it can attach it to the current process (even if `USE_DEBUGGER` has not been set). For example:

`setenv DEBUGGER /opt/gdb/6.8/bin/gdb` (where the latest version of GDB is specified)

In case of application crash, the crash detector also checks for the sender of the kill signal. If the signal is sent by any user, then the crash detector is triggered only when the following variable is set:

```
$CDS_ERRORLOG="ALL_ERRORS=YES"
```

Also, in the `CDS.log` file, one of the following statements get appended:

- Process was terminated by USER with #### signal

  When the process is terminated by the user

- Process was terminated with #### signal

  When the process is terminated by the system

You can run `cdsLibDebug` and `cdsinfo` commands to add more information in the generated crash report.

The `cdsLibDebug` command enables you to get a list of `cds.lib` files and library definitions. In addition, it enables you to find syntax errors in `cds.lib` files.

Whereas, the `cdsinfo` command provides detailed information on design libraries, such as `DMTYPE` or `NAMESPACE`. The `DMTYPE` library is used with design management system and `NAMESPACE` is used with `CDBA` or `OA`.

Crash Trend Reporting

Important

Cadence recommends that session logging information only be used on a local, rather than global, site basis to ensure that Virtuoso performance is not impacted.

You can track session start and end data using the name-value pair `STARTINFO_FILE` with the `CDS_ERRORLOG` environment variable (see Customizing Crash Reports).

After application exit, whether as a result of a crash or normal exit, a range of system data can be collected. This information can then be analyzed to measure trends, for example, in the number of session starts against the number of system crashes.

If the name-value pair `STARTINFO_FILE` is set to point to a writable file, each Virtuoso session will be logged.

Tip

This log file can however greatly increase in size following numerous recordings of start/stop sessions. It is therefore suggested that you consider making these files date dependent on a daily, weekly, or monthly basis. For example, `/tmp/startinfo_07_2009`. You can set the file name to automatically change using the following: `setenv CDS_ERRORLOG "STARTINFO_FILE=/home/jsmith/startTimes_`date +%b%Y`"`.

This log will include details of the program version, process ID, username, start/end dates and also times, or, in the case of a crash, the start/crash date and time will be recorded. Details of the process ID and crash report file name are also included. This can be either an absolute file path or the filename (which can be created and updated using the name-value pair `DIR` (see Saving Crash Report Data) or `/tmp`).

For example:

```
[ 7555@mymachine@jsmith ] start date/time:Sat Mar 21 06:37:57 2012 virtuoso
ICADV12.1.155 plat:sun4v
crash report file :: /tmp
crashReport_032109_0638_ICADV12.1.155_jsmith_mymachine.log
```

STARTINFO_FILE contains the name of the “startinfo file” under CDS_ERRORLOG (or under /tmp if DIR is not set). The “startinfo file” is created automatically if it does not exist, or it can point to an existing file.

Note: If you shutdown a virtuoso process using the `kill -9` command, the STARTINFO_FILE will not record the exit/crash.

Note: To send the stack trace to the log file when the application hangs, type the following command in the UNIX shell terminal:

```
kill -s USR1 <pid>
```

Availability of Signal Description Information for Crashes

The crash detector report also displays the termination signal name and number, for example “`signal:Hangup(1)`”. This provides additional information on the type of crash that has occurred, such as internal application errors recorded as, for example, “`#11 (segFault)`” or “`#4 (illegal instruction)`” .Alternatively, it can inform if the software has likely been killed by the user, for example “`#15 (sigterm)`” and “`#6 (sigabort)`”.

If a signal description is therefore available, to describe the type of crash, the STARTINFO_FILE will display it as follows:

```
[7555@mymachine@jsmith] 11(SegFault):Wed Jul 8 13:16:03 2012 virtuoso
ICADV12.1.2.404 lnx86
```

It should be noted however that signal numbers and descriptions may vary from platform to platform.

Storing User Feedback on Crashes

When a Virtuoso session crashes, the Fatal Application Error dialog appears. In this dialog, you can specify the nature of the Virtuoso crash. This information is stored in the crash report and by STARTINFO_FILE.

For example, if you select the option *Virtuoso unexpectedly crashed*, STARTINFO_FILE stores this information in the following format.

```
[7555@mymachine@jsmith] crash(Aborted):Fri Aug 12 15:52:08 2011 Virtuoso
ICADV12.1.6-64b.DEL.173 lnx86 ****PROCESS CRASHED****
```

Crash Due to X Server Error

If a crash occurs due to a X Server error, the startinfo file will use the term "`xkill`" to distinguish this type of crash (rather than "`crash`"). For example:

```
[ 7555@mymachine@jsmith ] xkill date/time:Sat Apr 21 06:42:46 2012 virtuoso
ICADV12.1.4.375 plat:sun4v
```

Crash Detector Display Check

The crash detector also performs a response check of `XtOpenDisplay()` on the machine being used for software display, recording the setting at the time of crash.

If this check establishes that `XtOpenDisplay()` is not responding, it will return `NULL`, and output that information into the X server section of the crash report. For example, `(xserver: (Not Responding)`).

Note: If `virtuoso` is being run in `-nograph` mode this check will not be performed.

Standard Virtuoso Exit

If Virtuoso exits as expected, that is it has not terminated as the result of a crash or X Server error, the following information format will be displayed in the `STARTINFO_FILE`:

```
[7555@mymachine@jmisth] exit(111):Thu Jun 11 12:35:43 2012 virtuoso
ICADV12.1.4.387 sun4v
```

Reporting Additional Crash Data

If you require additional crash data to be collected you can set the `EXTENDED` name-value pair for the `CDS_ERRORLOG` environment variable (see Customizing Crash Reports) to be `TRUE`.

If `EXTENDED` is activated, the crash report will also record details of your environment variable settings and the content of the `.cdsinit/cdsenv` files.

- Where for `.cdsinit`, the content of the `.cdsinit` file will be included in the crash data.
- Where for `.cdsenv`, the name/value of those cds environment variables that differ from their default values will be included in the crash data.

It will also report `checkSysConf` and `xdpyinfo` information.

- Where for `checkSysConf`, the results of checks on the necessary pre-requisites to run Cadence software on the machine that the crash took place will be reported.
- Where for `xdpyinfo`, the results of checks on the available graphics display visuals will be reported.

Note: `EXTENDED` is set to `TRUE` by default.

Saving Crash Report Data

Crash reports are saved into the directory specified by the `DIR` name-value pair using the `CDS_ERRORLOG` environment variable (see also Customizing Crash Reports). If `DIR` has not been specified, the directory name is generated by `cdsGetTmpDir()`.

Note: If you have specified a directory that does not currently exist, this directory will automatically be created.

Each crash report has the default format name:

"`crashReport_<DATE>_<TIME>_<SUBVERSION>_<USER>_<HOST>.log`"

This can however be customized by setting the `NAME_FMT` name-value pair which defines the generated report name. The following keywords are used to achieve this:

`%DATE` = this is substituted by the current date
`%TIME` = this is the current time
`%VERSION` = this is the application version
`%SUBVERSION` = this is the application subversion
`%PID` = this is the process ID
`%USER` = this is the user name
`%HOST` = this is the host name

For example:

```
CDS_ERRORLOG='NAME_FMT=crashReport_%DATE_%PID_%SUBVERSION_%USER_%HOST.log'
```

could generate a report name of:

```
/opt/reports/
crashReport_180709_1014_ICADV12.1.4.302_jsmith_msc080.log
```

Note: `DIR` can be set as an absolute path or as a path relative to the application startup directory.

For example:

```
setenv CDS_ERRORLOG 'DIR=/home/jsmith/errorlog_dir'

setenv CDS_ERRORLOG 'DIR=../<dir_name>/errorlog_dir'
```

There might be instances where several crash reports are generated at the same time and the timestamp is same in all these report. In such a scenario, the newly-generated crash report name is appended by a unique identifier. For example, *<generated-crash-*

report-name>_1.log, <*generated-crash-report-name*>_2.log ... <*generated-crash-report-name*>_N.log, to avoid overwriting existing reports.

Measuring Graphics Performance

Graphical performance measurements, which can be used to evaluate the ability of a platform to run the various graphics operations required by Virtuoso applications, can be accessed by running the `hiGraphicsBenchmark` standalone application from a shell terminal. This application is located in `/tools/dfII/bin` in your `virtuoso` installation.

Running `hiGraphicsBenchmark` displays the Virtuoso Graphics Performance Benchmarks window (see Running Virtuoso Graphics Performance Benchmarks).

Note: Examples of graphic performance measurements include on-screen redraw performance, area copy performance, and image operation performance. For more information about all of the test measurements available, see Performance Benchmarks.

Using graphic performance measurements can provide a number of benefits, including:

- Establishing whether your current system can comfortably run `virtuoso`
- Tracking a change in performance following any system updates, such as installing a new driver
- Locating bottlenecks or issues which can then allow for more accurate system remedies to be provided
- Providing suggestions, hints, and settable options that you can use to improve graphics performance with `virtuoso` (see the Calibrator)

Running Virtuoso Graphics Performance Benchmarks

1. Run `hiGraphicsBenchmark` from a shell terminal.
2. In the displayed Graphics Performance Benchmarks window, select the benchmark tests that you want to perform.
3. Click the *Run* button, or choose *File – Run Selected Benchmarks* to begin testing.

   Results are displayed in the Benchmark's Results Window.
4. Optionally, run the *Calibrator* (graphics measurement test tool) by clicking the *Run* button, or by choosing *Tools – Calibrator*.

   Note: You can run the Calibrator on a set of pre-selected benchmarks. The results of calibration are displayed as suggested improvements in the Calibration and Suggestion Window.
5. Optionally, open the Task Viewer to view the system processes that are currently running and the current levels of CPU memory usage.

hiGraphicsBenchmark Command-Line Arguments

The following command-line arguments can be used with `hiGraphicsBenchmark`:

`[-taskviewer [PID]]`	Runs `hiGraphicsBenchmark` as a task viewer. Optionally, a process `PID` can be specified to monitor its CPU usage, memory usage and so on. If `PID` is not specified, it will search for any `virtuoso` process, or use the current process as default.
`[-calibrator]`	Runs `hiGraphicsBenchmark` as a calibrator to test key graphics operations, and provide alternative solutions, through settings or command line operations so as to improve graphics performance.
`[ -h | -H]`	Displays `hiGraphicsBenchmark`.

Graphics Performance Benchmarks Window

The Graphics Performance Benchmarks window is displayed when you run the `hiGraphicsBenchmark` command from a shell terminal.

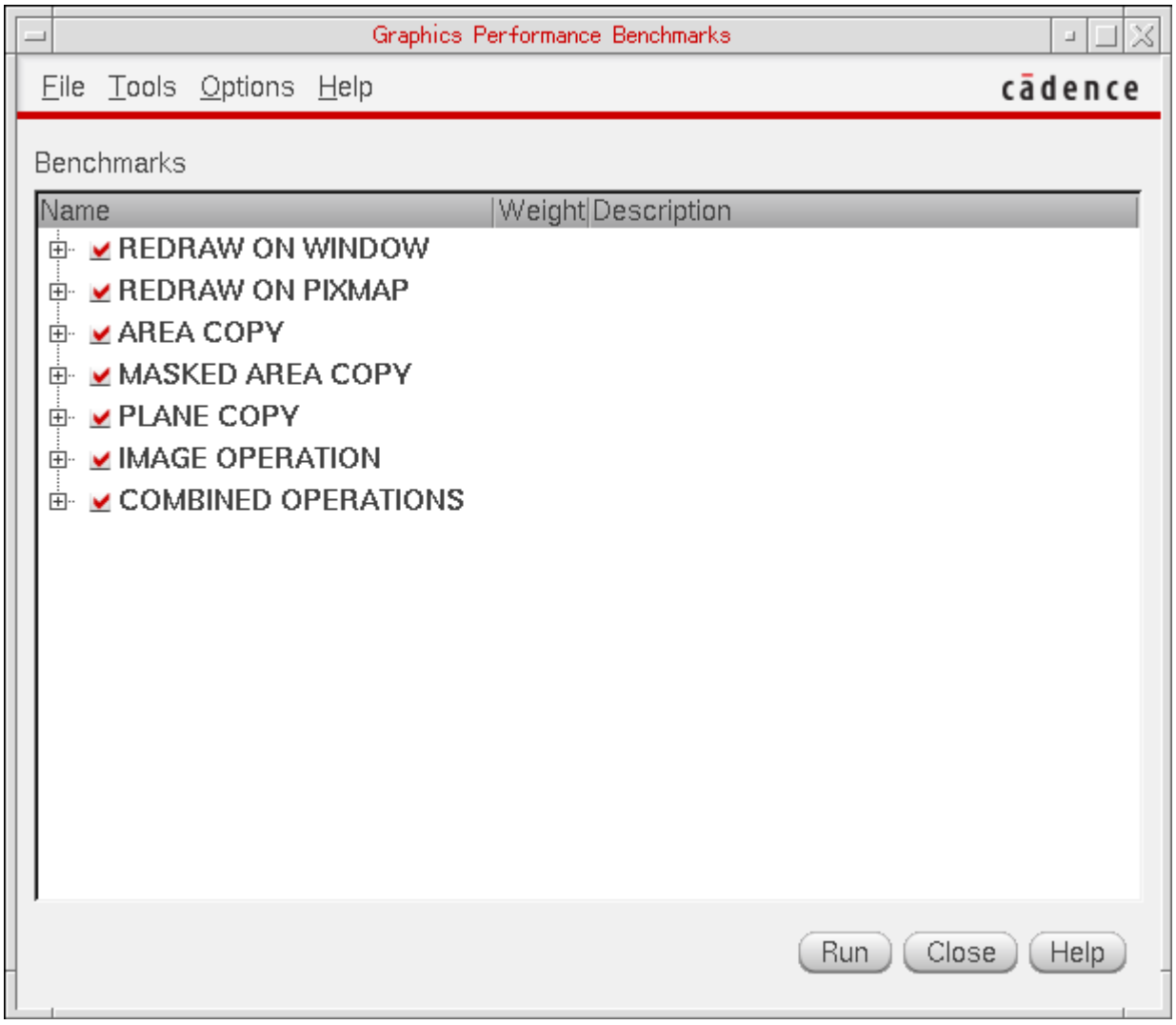
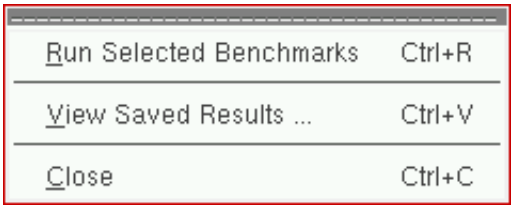
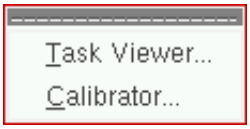
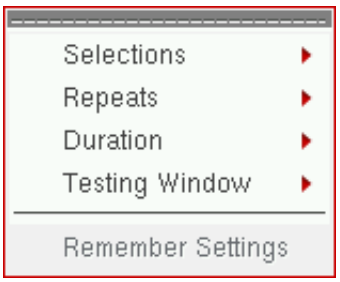


Figure D-4 The Virtuoso Graphics Performance Benchmarks Window

The Virtuoso Graphics Performance Benchmarks window comprises the following elements:

Option	Description
File menu 	Choose *Run Selected Benchmark* to run the benchmark tests that are currently selected in the *Benchmarks* table. This is equivalent to using the *Run* button. Choose *View Saved Results* to access the Open form from where you can open and view previously saved results. Choose *Close* to exit the Graphics Performance Benchmarks window.
Tools menu 	Choose the appropriate command to open the Task Viewer window or to run the Calibrator.

Option	Description
Options menu  	Choose: ■ *Selections* to filter the benchmark tests to be performed. For example, select *All* tests, or only those tests that are part of the *Redraw Set*. ■ *Repeats* to specify how many times the selected tests must be run. For example, if you have selected the *REDRAW ON WINDOW* group, and have set *Repeats* to *Twice* (the default), the test group will be run twice. Using *Repeats* can increase testing accuracy because the program keeps track of the number of times the tests are run and reports an average result. ■ *Duration* to specify the time for which each test must run. For example, if you set the *Duration* to 2 seconds (the default), each selected test runs for two seconds, performing the same check operation until the time set is reached. The higher the value that is set, the higher is the testing accuracy. ■ *Testing Window* commands to specify how the window is to be displayed. The default is to display the window *With Frame*. With a frame, you can perform operations such as moving the window and changing the window stack (through window management). The name of the current test will also be displayed in the frame to monitor test progress. Without a frame, the testing window is always displayed on top, with window management playing no role, and consequently it does not interfere in the test. ■ *Remember Settings* to save the current settings to the `.benchmark` file in your home directory. Saved values are used as the default values the next time the program is run.

Option	Description
Benchmarks table	Displays information about all the available graphics performance tests (*Name* column) that can be run. Select the check box corresponding to a test if you want to run that test. Tests are grouped into specific testing areas, such as *REDRAW ON WINDOW*, *AREA COPY*, and *IMAGE OPERATION*. For more information on these tests, along with a description of each, see Performance Benchmarks. A *Weight* and *Description* can also be provided for each benchmark test. To edit the weight (importance) to be applied to a test, double-click the value in the *Weight* column and type the new value, or use the up and down arrows.

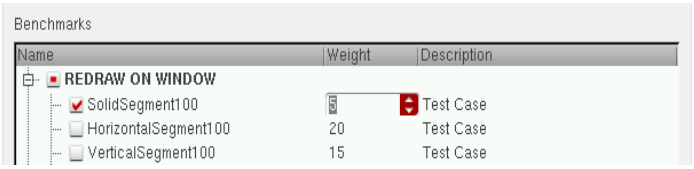


Figure D-5 Editing a Test Weighting

Run	Runs the selected tests (displaying the Testing window). The *Run* button becomes available only if one or more specific benchmark tests are selected in the *Benchmarks Table*. **Note:** After testing starts, the *Run* button changes to a *Stop* button to enable you to interrupt the testing, if required. Test results are displayed in the Benchmark Results window.
Close	Exits the Graphics Performance Benchmarks window.

Performing Benchmark Tests

There are two methods that can be used to perform benchmark tests.

1. To run the benchmark tests that are currently selected in the Graphics Performance Benchmarks window, click the *Run* button, or choose *File – Run Selected Benchmarks*. A Testing window is displayed, and after the testing is complete, the results are displayed in the Benchmark Results window.

2. You can also run the *Calibrator* tool by selecting *Tools – Calibrator* to perform a group of predefined benchmark tests designed to measure performance of certain operations that are specific to `virtuoso`, such as blinking.

   To run the Calibrator tool, you need not select any specific benchmark tests from the *Benchmark* table. You can monitor the progress of the calibration tests in the Testing window while the tests are being run. After the calibration is complete, the results are displayed in the Calibration & Suggestion window. The Calibration & Suggestion window

provides general test analysis results, along with suggestions on how to improve system performance in relation to graphics.

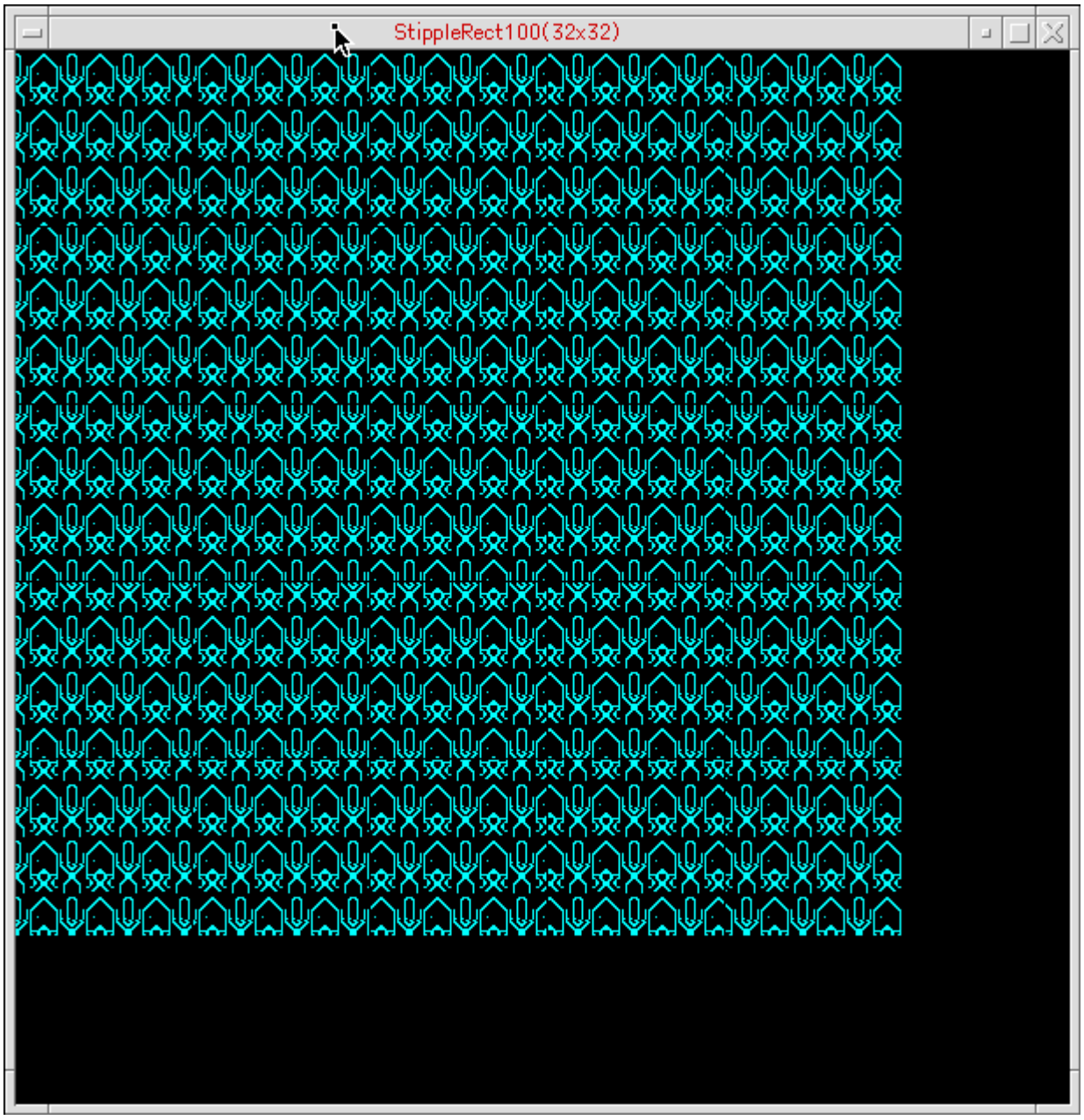


Figure D-6 Graphic Benchmarking Being Performed in the Testing Window (Currently Testing SolidRect50)

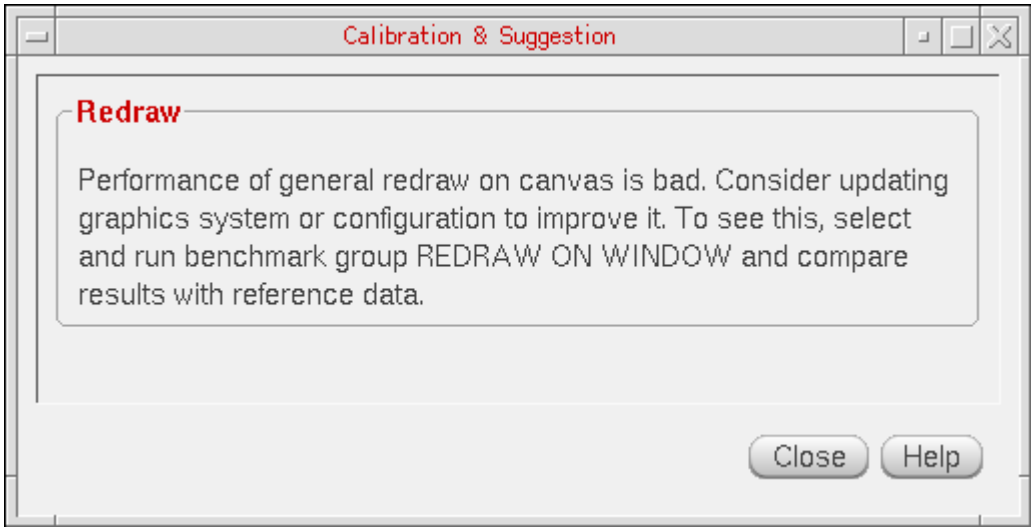


Figure D-7 Calibration Results and Suggestions to Improve Performance

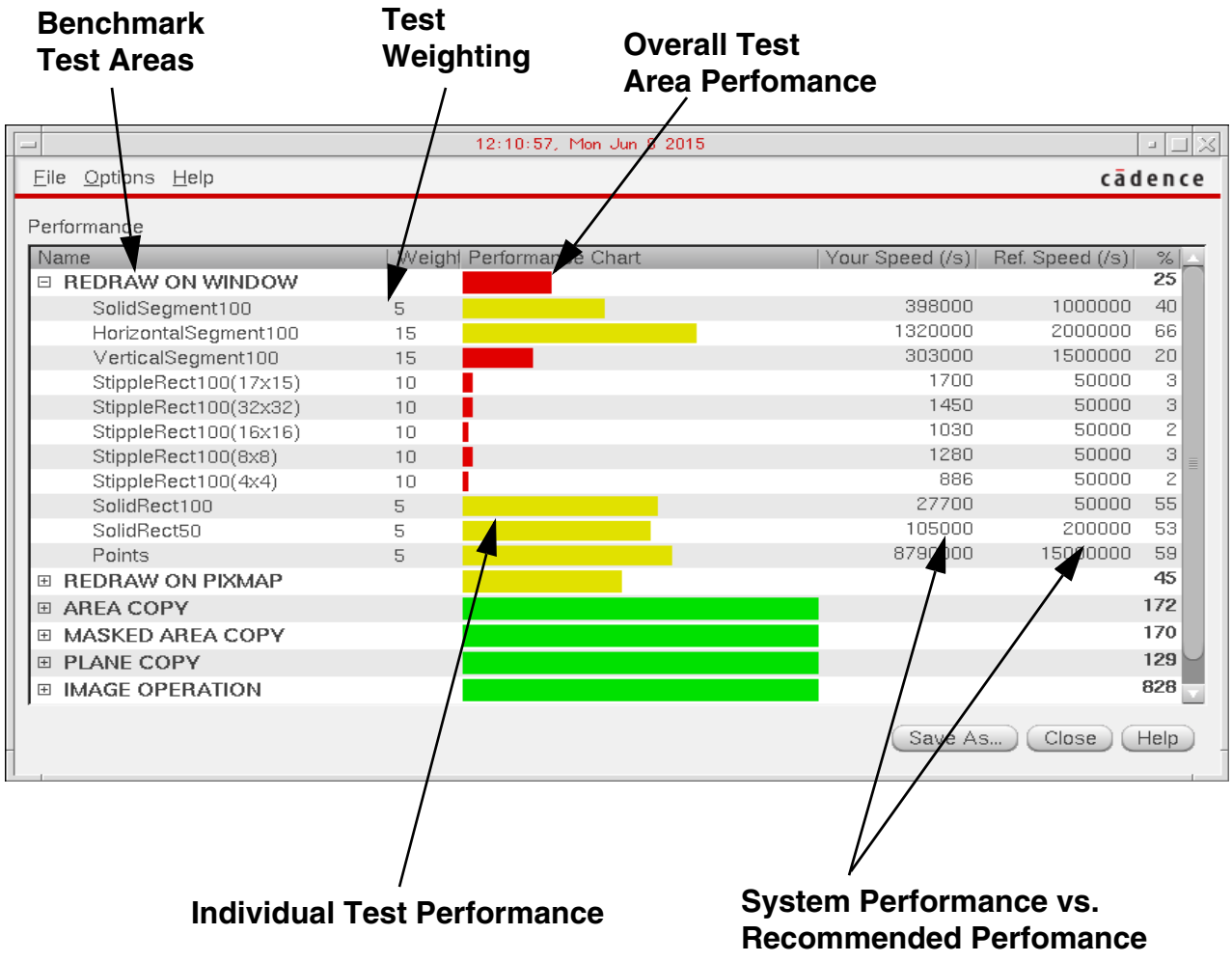


Figure D-8 Benchmark Results Window

The Benchmark Results window contains the following information columns:

Column	Description
Name	Details the benchmark test names. If a benchmark test was not run, it is grayed out in the window and is also not displayed in the *Options* menu.
Weight	The weight of an individual benchmark that is used to measure its contribution to the performance of its benchmark group.

Column	Description
Performance Chart	Graphical representation of performance. Each individual benchmark test has its own performance measure, while the group performance measurement is a weighted average of all of the selected benchmarks in that group. With the color scheme currently selected in the figure above, *Green, Yellow,* and *Red* are used to relay *Performance Chart* results. If required, you can change the color scheme used by choosing *Option – Color Code*. Additionally, you can use the *Options* menu to change the results display from a *2D Bar* to a *3D Bar Chart Presentation*, and also to choose whether to display *All Benchmarks* or *Only Selected Benchmarks*. After any changes have been made to the window settings, you can save them for reuse by choosing *Option – Remember Settings*.
Your Speed (/s)	The absolute performance for a benchmark test.
Ref Speed (/s)	The collective performance that represents a reference standard that is deemed “good enough” for smooth performance.
%	The ratio of *Your Speed* against *Ref Speed*.

You can save the results in a test (`.tst`) file by choosing *File – Save As*. A test results file can then be reloaded in the Graphics Performance Benchmarks window by choosing *File – View Saved Results*.

Important

To obtain the best calibration results, and related suggestions to improve performance, it is recommended that you use a combination of calibration run methods, amending the selected tests.

Using The Task Viewer

Important

The Task Viewer tool is currently available only on Linux.

In the Graphics Performance Benchmark window, choose *Tools – Task Viewer*, in the Virtuoso Graphics Performance Benchmarks window, to display the Task Viewer window.

Note: The *Task Viewer* automatically updates after every 2 seconds.

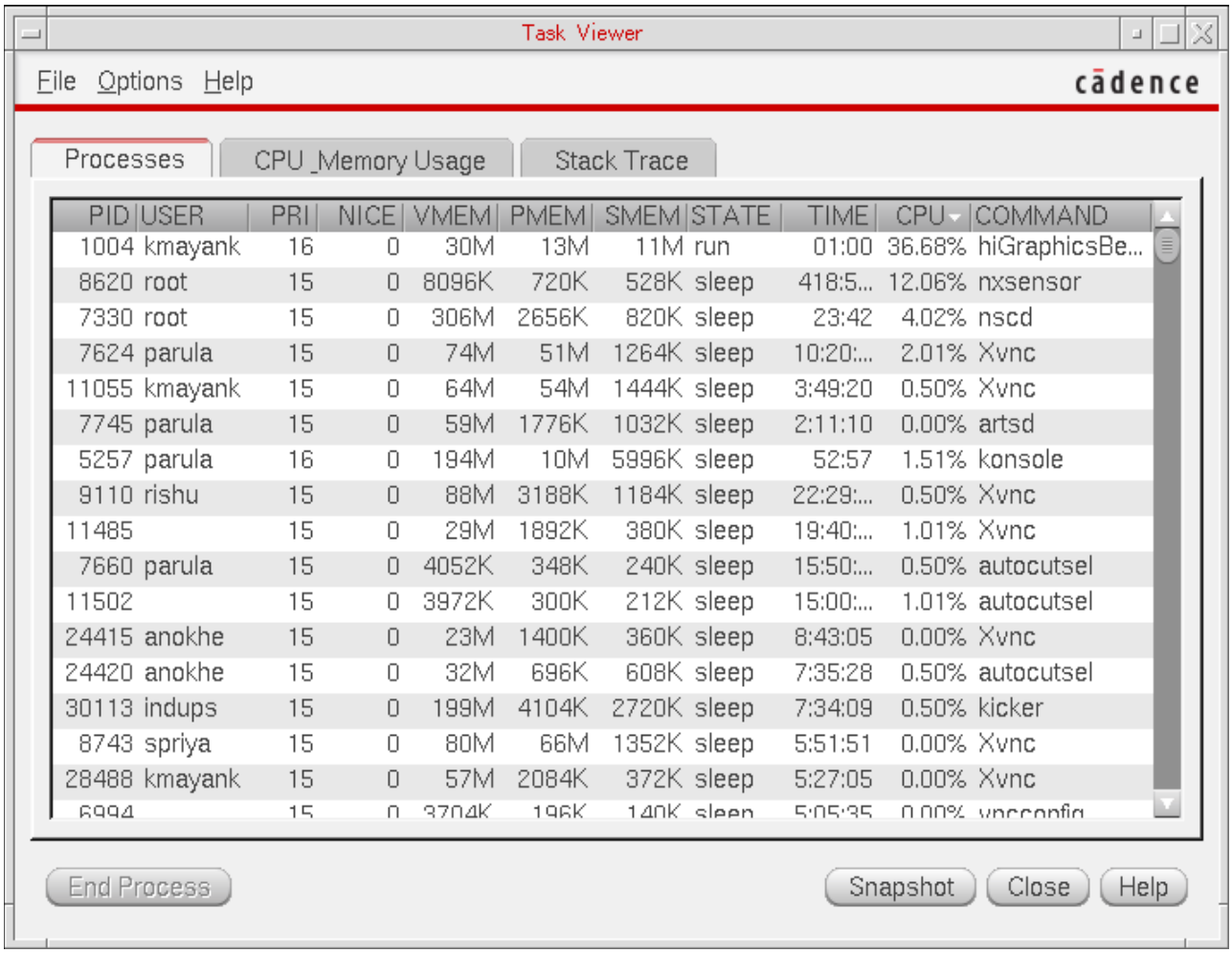


Figure D-9 The Processes Tab in the Task Viewer

In the Task Viewer, you can view the *Processes* that are currently being run to determine which processes will impact performance analysis results, and also the *CPU_Memory Usage*, *X Resources*, and the *Stack Trace*.

Note: The information contained on the *Processes* and *X Resources* tabs can be sorted (the `virtuoso` and `X` resource processes are always highlighted).

Additional features of the *Task Viewer* include:

- The ability to end or kill a process in the *Processes* tab by selecting the process ID (PID) of the required process, and then selecting the *End Process* button.

  Note: The *End Process* button is active only when a process is selected in the *Processes* tab.

- The ability to capture the information displayed by taking a *Snapshot*. You can use a snapshot for reporting purposes.

  When a snapshot is taken, a time tag is attached to the file name so that you can determine when the snapshot was taken (for example, `snapshot@16:52:05`).

  To save the snapshots, choose *Options – Set Snapshot Folder/Name* and then specify the name and storage location in the displayed Set Snapshot Folder and Name form.

  Note: The snapshot file is saved in the ASCII format and contains the following details:

  - All the processes listed on the *Processes* tab, with their current states.
  - The stack trace at the moment the snapshot was taken.
  - The X server configuration file, if it exists.
  - The X server log file, if it exists.
  - The X display information.

- The ability to set the *Memory Unit* displayed on the *CPU_Memory Usage* tab.

  Choose *Option – Memory Unit*, and then choose to display results in either *2G* (default) or *1G* unit format.

  Note: The *Memory Unit* option is active only when the *CPU_Memory Usage* tab is selected.

- The ability to record preferred settings by choosing *Option – Remember Settings*.

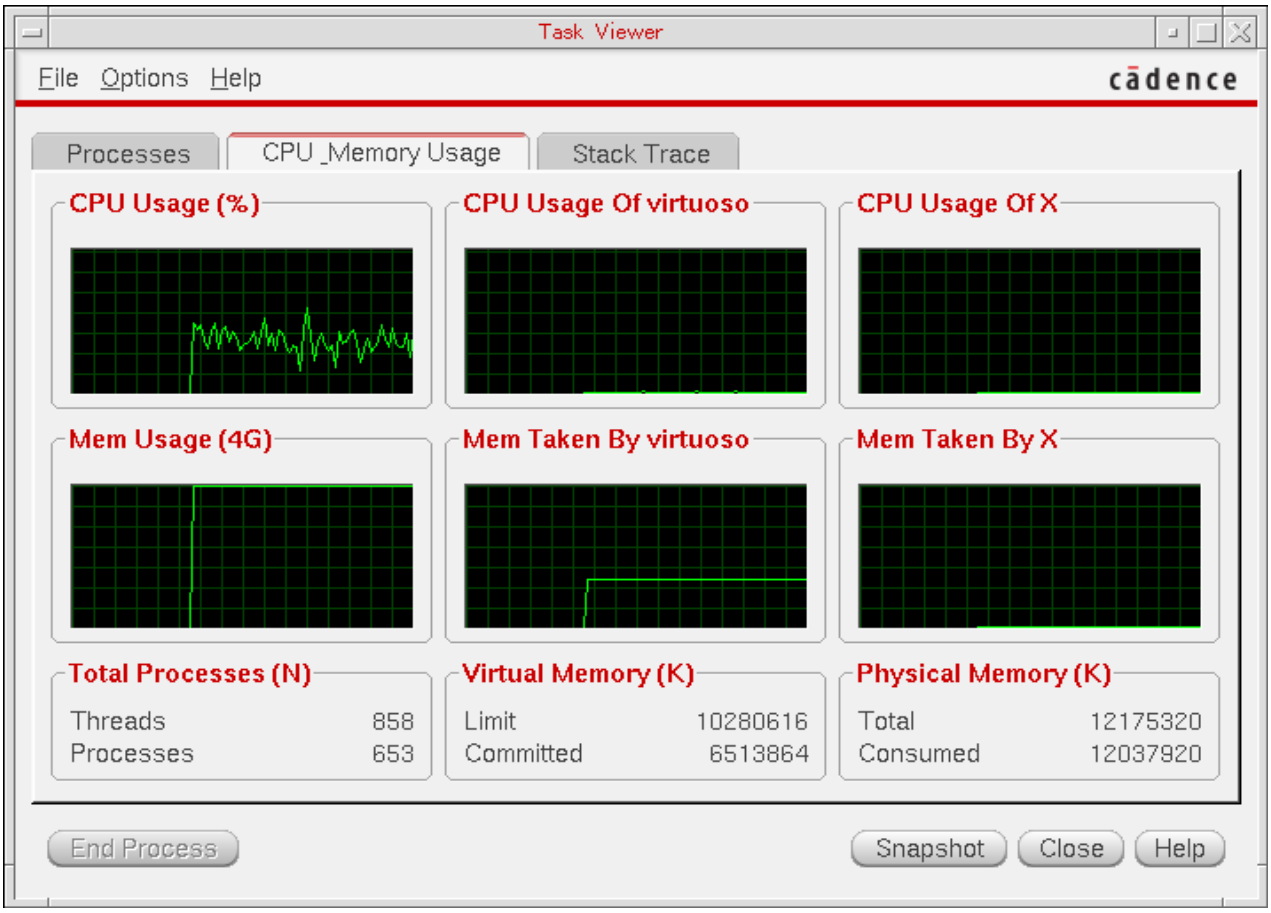


Figure D-10 The CPU_Memory Usage Tab in the Task Viewer

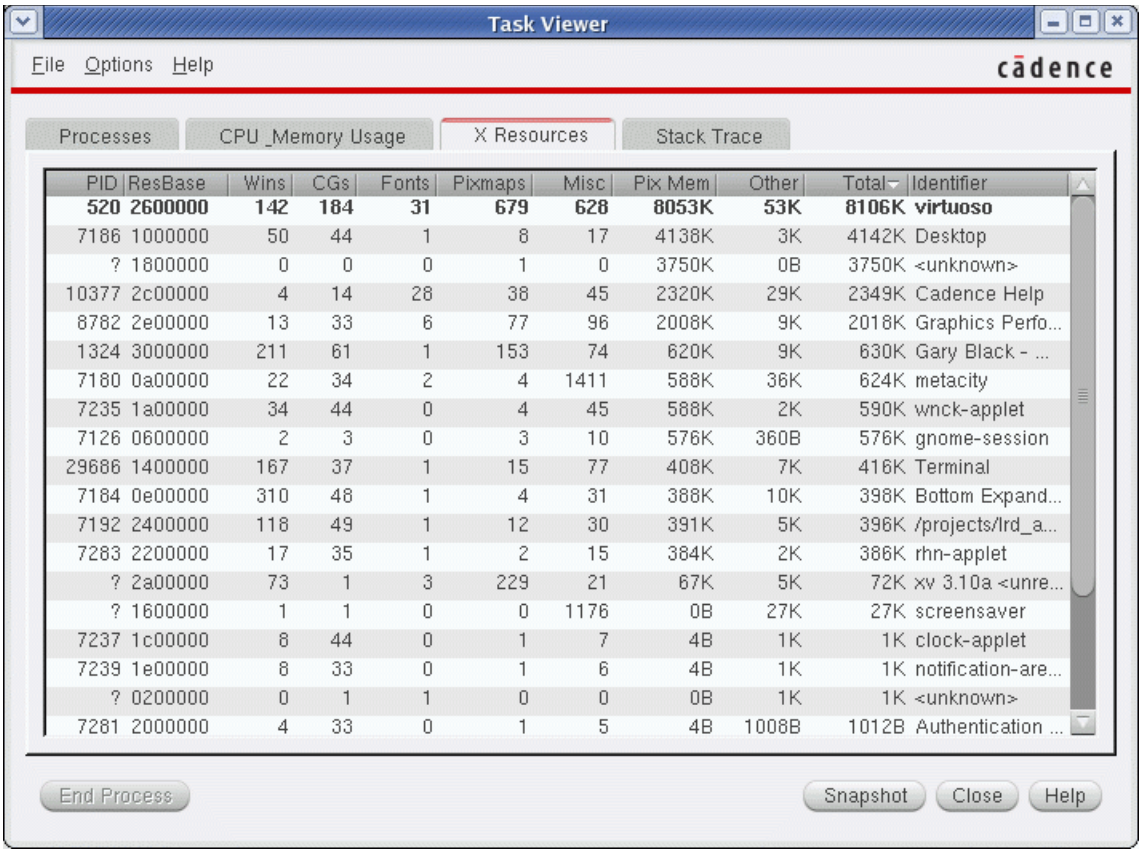


Figure D-11 The X Resources Tab in the Task Viewer

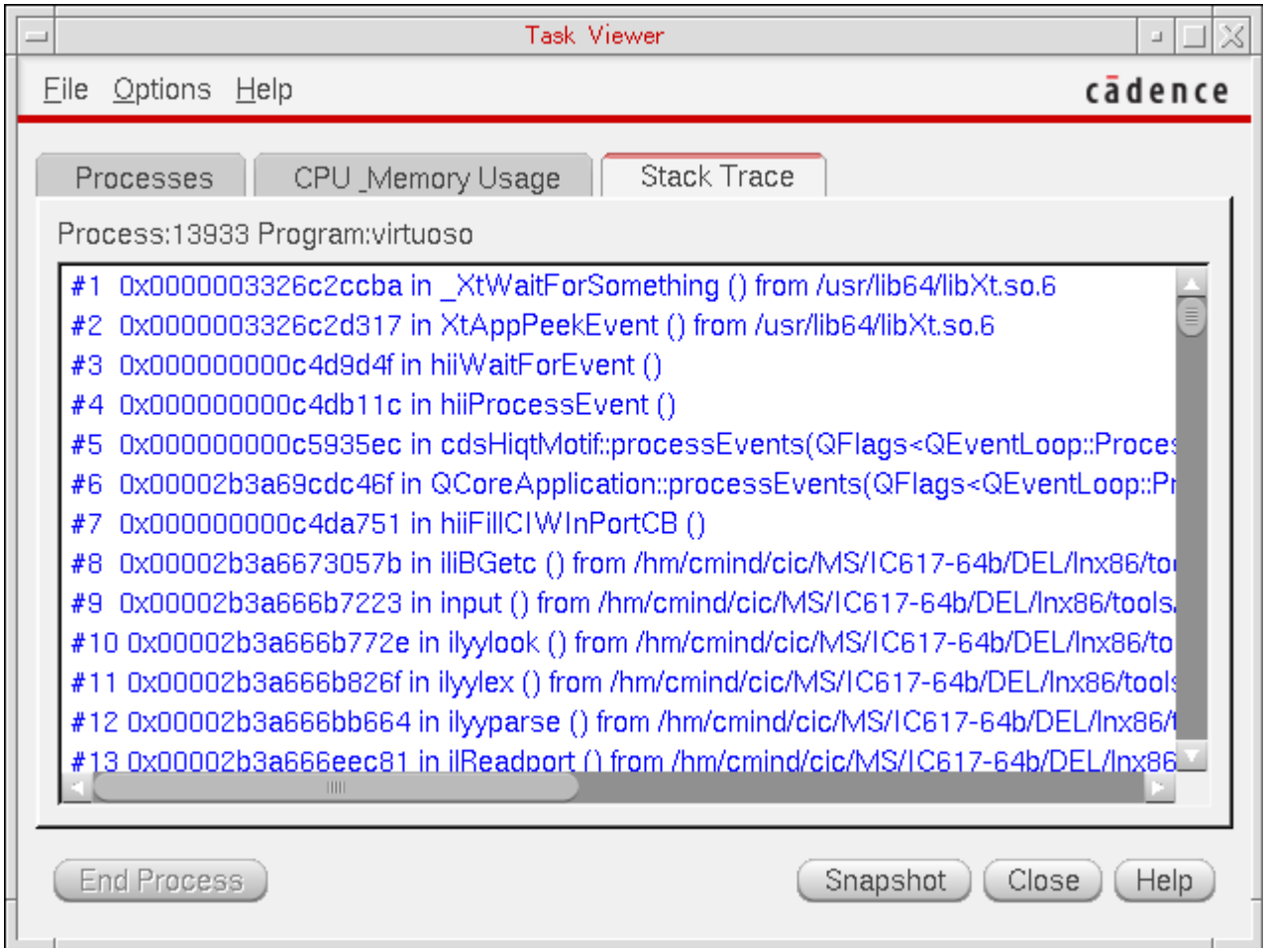


Figure D-12 The Stack Trace Tab in the Task Viewer

Performance Benchmarks

REDRAW ON WINDOW

Benchmark	%	Description
SolidSegment100	5	random lines in length of 100 pixels
HorizontalSegment100	15	horizontal lines in length of 100 pixels
VerticalSegment100	15	vertical lines in length of 100 pixels
StippleRect100(17x15)	10	stippled rectangle with size of 100 pixels and a 17x15 stipple
StippleRect100(32x32)	10	stippled rectangle with size of 100 pixels and a 32x32 stipple
StippleRect100(16x16)	10	stippled rectangle with size of 100 pixels and a 16x16 stipple
StippleRect100(8x8)	10	stippled rectangle with size of 100 pixels and a 8x8 stipple
StippleRect100(4x4)	10	stippled rectangle with size of 100 pixels and a 4x4 stipple
SolidRect100	5	solid rectangle with size of 100
SolidRect50	5	solid rectangle with size of 50
Points	5	2048 random points

REDRAW ON PIXMAP

Benchmark	%	Description
SolidSegment100	5	random lines in length of 100 pixels
HorizontalSegment100	20	horizontal lines in length of 100 pixels
VerticalSegment100	15	vertical lines in length of 100 pixels
StippleRect100(17x15)	10	stippled rectangle with size of 100 pixels and a 17x15 stipple
StippleRect100(32x32)	10	stippled rectangle with size of 100 pixels and a 32x32 stipple
StippleRect100(16x16)	10	stippled rectangle with size of 100 pixels and a 16x16 stipple
StippleRect100(8x8)	10	stippled rectangle with size of 100 pixels and a 8x8 stipple
StippleRect100(4x4)	10	stippled rectangle with size of 100 pixels and a 4x4 stipple
SolidRect100	5	solid rectangle with size of 100
SolidRect50	5	solid rectangle with size of 50

PLANE COPY

Benchmark	%	Description
Pixmap2Bitmap500	40	from pixmap to bitmap in size of 500
Bitmap2Pixmap500	5	from bitmap to pixmap in size of 500
Bitmap2Window500	5	from bitmap to window in size of 500
Pixmap2Bitmap200	40	from pixmap to bitmap in size of 200
Bitmap2Pixmap200	5	from bitmap to pixmap in size of 200
Bitmap2Window200	5	from bitmap to window in size of 200

AREA COPY

Benchmark	%	Description
Pixmap2Pixmap500	20	pixmap to pixmap copy in size of 500
Pixmap2Window500	15	pixmap to window copy in size of 500
Window2Pixmap500	10	window to pixmap copy in size of 500
Window2Window500	5	window to window copy in size of 500
Pixmap2Pixmap200	20	pixmap to pixmap copy in size of 200
Pixmap2Window200	15	pixmap to window copy in size of 200
Window2Pixmap200	10	window to pixmap copy in size of 200
Window2Window200	5	window to window copy in size of 200

MASKED AREA COPY

Benchmark	%	Description
Pixmap2Pixmap500M	20	pixmap to pixmap copy in size of 500
Pixmap2Window500M	15	pixmap to window copy in size of 500
Window2Pixmap500M	10	window to pixmap copy in size of 500
Window2Window500M	5	window to window copy in size of 500
Pixmap2Pixmap200M	20	pixmap to pixmap copy in size of 200
Pixmap2Window200M	15	pixmap to window copy in size of 200
Window2Pixmap200M	10	window to pixmap copy in size of 200

Benchmark	%	Description
Pixmap2Pixmap500M	20	pixmap to pixmap copy in size of 500
Window2Window200M	5	window to window copy in size of 200

IMAGE OPERATION

Benchmark	%	Description
BlendImage500	10	alpha blending in size of 500
SmhBlendImage500	10	alpha blending in size of 500 with XSHM
PutImage500	20	copy image to window in size of 500
SmhPutImage500	20	copy image to window in size of 500 with XSHM
GetImage500	20	grab image from window in size of 500
ShmGetImage500	20	grab image from window in size of 500 with XSHM

COMBINED OPERATIONS

Benchmark	%	Description
ColorDrag400	15	color dragging of a 400x400 rectangle
ColorDrag200	15	color dragging of a 200x200 rectangle
ColorStretch	10	color stretching with a rectangle
MonoColorDrag400	10	mono-color dragging of a 400x400 rectangle
MonoColorDrag200	10	mono-color dragging of a 200x200 rectangle
MonoColorStretch	10	mono-color stretching with a rectangle
Blinking500	10	blinking operation with X calls
Blinking500Hi	10	blinking operation with Hi calls
ColorHighlight	10	highlighting operation

Acknowledgements

Cadence would like to acknowledge the following sources of information:

- `xllperf` (for testcase re-use)

  Copyright 1988-1989, Digital Equipment Corporation, Maynard, Massachusetts.

- `top` (for extensive code re-use in the development of the *Process* tab in the *Task Viewer*)

  Copyright 1984-2008, William LeFebvre.

- `xrestop` (for extensive code re-use in the development of the *X Resource* tab in the *Task Viewer*)

  Copyright 2003, Matthew Allum.

Using the Performance Diagnostic Tool

The Performance Diagnostic tool in the Virtuoso custom IC design platform contains many debugging techniques to isolate issues that might have caused an application to slow down or freeze. It has an auto-hiding control toolbar to let you record callstacks when the application slows down.

A callstack is a snapshot of the internal function calls during program execution. A series of callstacks within the issue duration (profiling) can give Cadence support team a sense of where the most time is spent.

The tool also analyzes the callstacks and creates a high-level summary. If the support team is not able to reproduce the issue based on your problem description, this summary will make it easier for the team to narrow down the causes or even provide an immediate workaround. The support team might ask further questions or provide instructions on using other diagnostic techniques in this tool.

The diagnostic process can be divided into three steps:

1. Installing the tool
2. Collecting Data
3. Reporting the Issues

Installing the tool

You can use one of the following ways to install this utility:

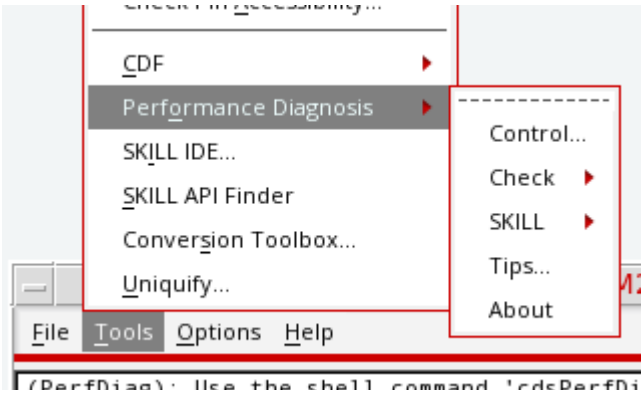
- Open the *Tools* menu in CIW and then choose *Performance Diagnosis – Install Submenu.*
- Set the following shell environment variable at the command-line:
  ```
  setenv CDS_PERFDIAG
  ```
- Launch the Virtuoso application with the `-perfdiag` flag:
  ```
  virtuoso -perfdiag
  ```
- Install the PerfDiagnosis control form by default by adding the following SKILL command to the `.cdsinit` file.
  ```
  perfDiagInstall(?openCallstack t)
  ```
  For more information, see `perfDiagInstall`.

Collecting Data

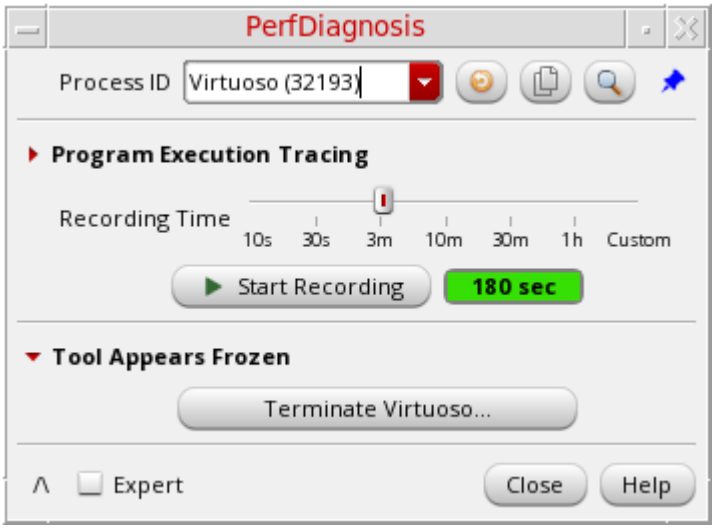
The PerfDiagnosis control form helps in collecting callstacks that later help the support team in identifying issues. You can pin this form at the top of your screen to save desktop space.

To start collecting performance data:

1. In CIW choose *Tools – Performance Diagnosis – Control.*



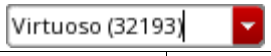
This will launch the PerfDiagnosis form.



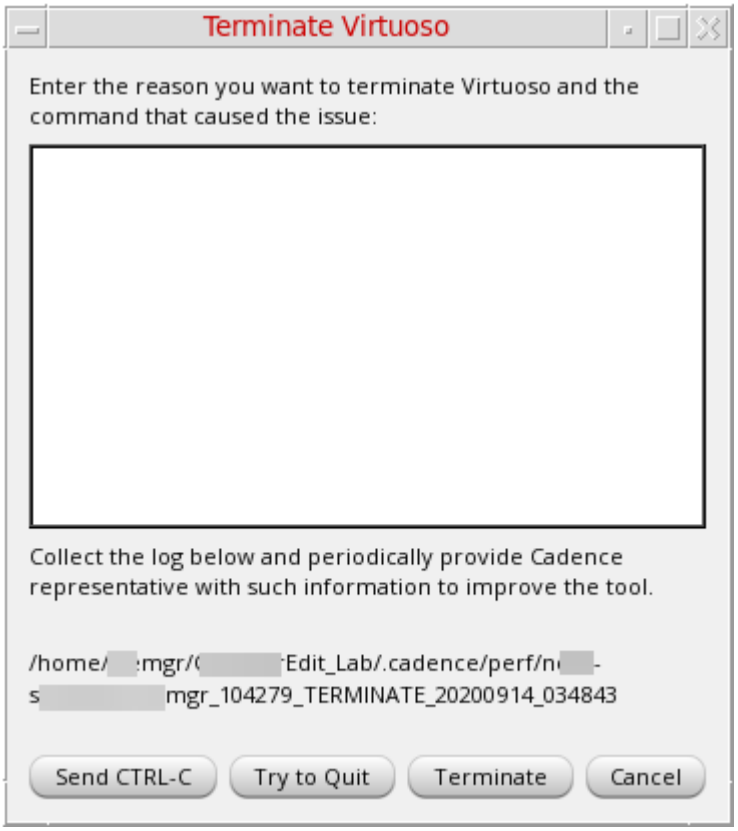
2. Adjust the recording time and then, click *Start Recording* when the issue appears. Click *Stop Recording* when you want to stop recording the callstacks.

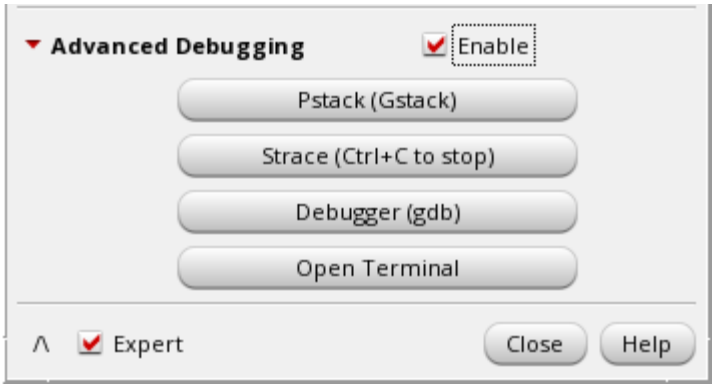
Options on the PerfDiagnosis Form

The following options in the PerfDiagnosis control form control how the data is collected:

Option	Description
Process ID	
	Select the process ID to track its performance.
	
	Refreshes the Process ID drop box list.
	Copies the process ID to the clipboard. This ID can be used in conjunction with UNIX commands in *Expert* mode.
	Select a frozen application window to bring up the PerfDiagnosis control form.
	Pins the minimized version of the form at the top of the screen.  When you point to this bar, the bar will expand to display commonly-used action buttons: *Expand, Start Recording*, *Stop Recording*, *Terminate Virtuoso*, *Show PerfDiagnosis control form* for the selected frozen window, and *Unpin the PerfDiagnosis* control form: 
Program Execution Tracing	
	Specify the time for recording the performance of the selected process ID.

Option	Description	
	Recording Time	Specify the time for which you want to collect the call-stacks for the specified *Process ID*. To do this, move the slider to the specified time on the *Recording Time* bar. If you move the slider to *Custom*, additional options are displayed to let you specify the *Interval* and *Time out* information. 
	Latency (sec)	Specify the time to wait before recording starts.
	Print All Threads	Prints all threads in the results. By default, only the main thread is printed.
	File Tags	Specify keywords to be appended to the log file name for categorization.
	Start Recording / Stop Recording	
		A toggle button that you can click to start recording the callstacks to collect the performance data of the specified process ID. The default recording time in 180 seconds. You can check the remaining recording time in the progress bar provide next tothe *Start Recording* button, or stop recording when the issue disappears.

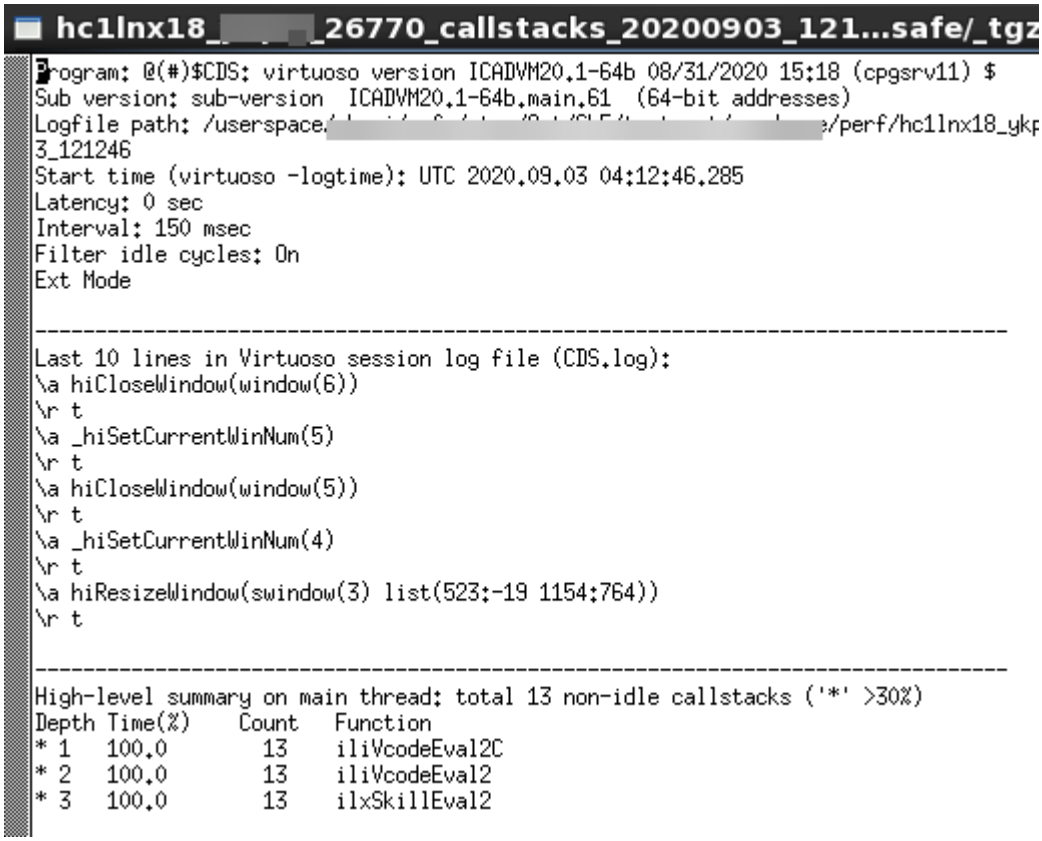
Option	Description
Tool Appears Frozen	
	This section provides options to terminate a frozen tool or application.
	Terminate Virtuoso
	Displays the Terminate Virtuoso form.  Click the *Send CTRL-C* button to interrupt the program causing the issue. If it is still unresponsive, specify the reason for exiting Virtuoso (for example, describe how you run into this issue) and then click the *Try to Quit* (possibly with a crashReport), or *Terminate* buttons. The Performance Diagnostic tool will record 10 sec of callstacks and close the current Virtuoso session.
Expert	Display the Advanced Debugging section.

Option	Description
Advanced Debugging	
	This section provides a UNIX expert more options to apply advanced debugging techniques. 
Enable	Enable the advanced debugging options.
	Pstack(Gstack)
	Invoke the `pstack` or `gstack` UNIX command to print a callstack of the specified PID
	Strace(Ctrl+C to stop)
	Invoke the `strace` UNIX command to trace system calls of the specified process ID.
	Debugger(gdb)
	Invoke the `gdb` UNIX command to attach a GNU Debugger to the specified process ID
	Open Terminal
	Opens a UNIX shell terminal window.

Reporting the Issues

When you stop collecting data, a log file is opened in a terminal window with the last 10 lines of `CDS.log` and a high-level summary of callstacks. It is recommended that you send the log to the Cadence support team along with your problem description or by at least writing down names of the functions on which the most time was spent. Functions with higher and lower

extremes are of particular interest. If the time taken is higher then it might be the cause of bottleneck.



Troubleshooting a Tool Freeze

To troubleshoot a sample tool freeze:

1. Run the following SKILL loop command from CIW:

   ```
   while(t nil)
   ```

2. In the PerfDiagnosis form, click *Start Recording*.
3. After recording for some time, click *Stop Recording*.
4. Click *Terminate Virtuoso*.

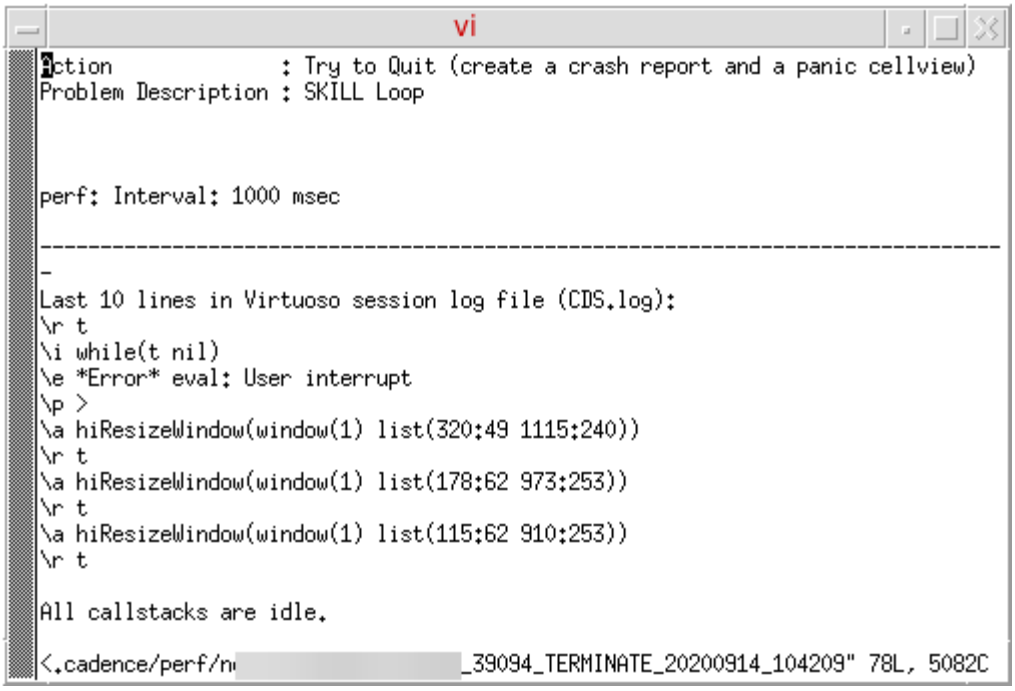
   The Terminate Virtuoso form appears.

5. To break the SKILL loop, click the *Send CTRL+C* button.

   When the SKILL loop is interrupted, the tool should be back to normal.

6. Repeat step 2 and if the tool freezes again, click *Try to Quit*.

   10 seconds of callstacks are collected and Virtuoso is terminated. In some cases, a crash report is also generated.



Checking Library Information

Conduct a thorough check of the design library access time, with multiple pings to each library server to assess average and worst case times. Depending on the number of libraries in the `cds.lib` file, this test might take several minutes.

➡ To perform this test, in CIW, choose *Tools – Performance Diagnosis – Check – Library.*

This option will open a terminal window with information about the current library. The results are shown in a viewer and are logged at `.cadence/perf/$HOST_$USER_$PID_checkLibrary_$time`

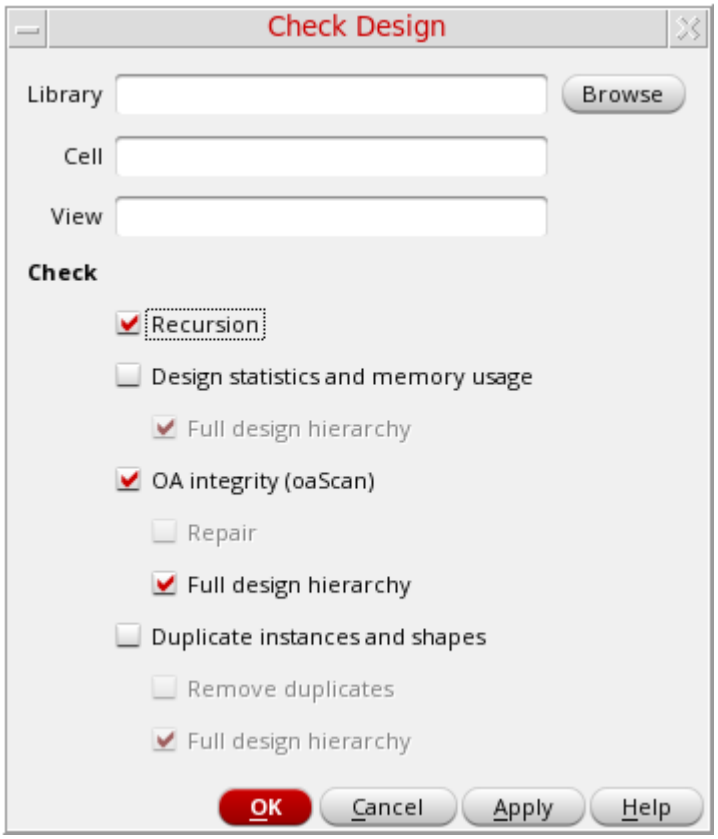
Note: The same functionality is supported in *Library Manager – Help – Diagnostics*.

Checking Design Information

➡ To check library-related information, in CIW choose *Tools – Performance Diagnosis – Check – Design.*

The Check Design form appears.

You can use the options on this form to run tests on the specified design.



The form contains the following options:

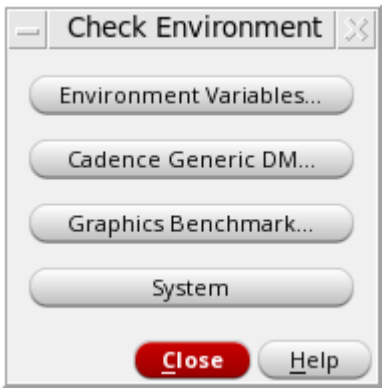
Option	Description
Library	Specify the library that contains the design that you want check. You can click the Browse button to select the library from Library Browser.
Cell	Specify the cell name.
View	Specify the view name.

Option	Description	
Check	Select the tests you want to run on the specified design.	
	Recursion	Checks the design hierarchy for recursive references (loops) that can cause a tool to freeze. These references need to be removed.
	Design statistics and memory usage	
		Checks the design statistics and find out how much memory is used by the design. ■ *Full design hierarchy* Select to run the check on the entire design hierarchy.
	OA Integrity (oaScan)	
		Runs the oaScan utility that checks for inconsistencies in the OpenAccess design, technology, and DMData databases and optionally repairs the inconsistencies and saves the databases. For more information, see Using the oaScan Utilities for Virtuoso. ■ *Repair* Select to repair any issues found during the oaScan. ■ *Full design hierarchy* Select to run the check on the entire design hierarchy.
	Duplicate instances and shapes	
		Checks the design to identify any duplicate instances or shapes in it. ■ *Remove duplicates* Select to delete any duplicate instances or shapes detected during the check. ■ *Full design hierarchy* Select to run the check on the entire design hierarchy.

Checking the Environment

➡ To check library-related information, in CIW choose *Tools – Performance Diagnosis – Check – Environment*.

The Check Environment form is displayed.

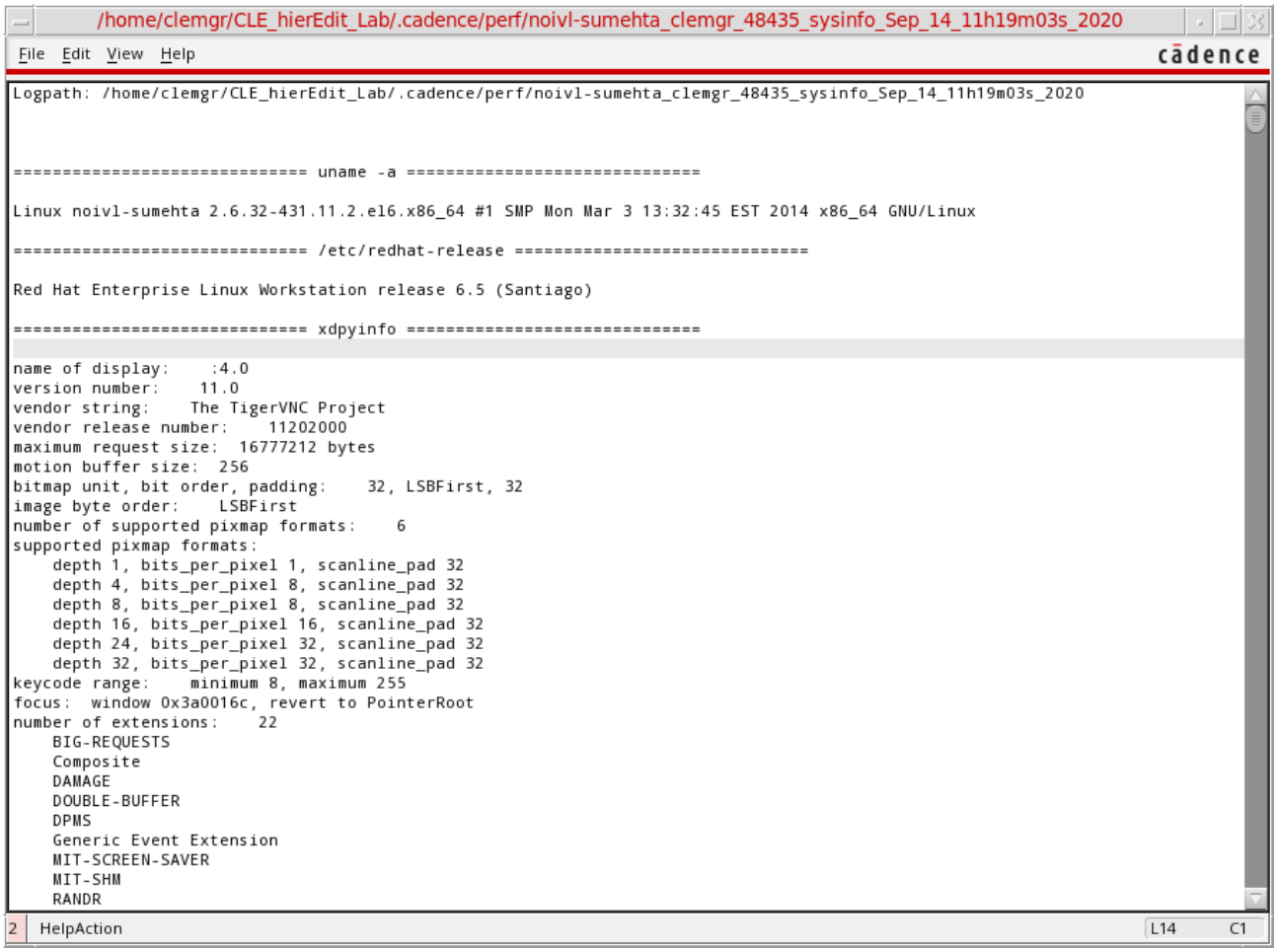


In this form, you can click the following options to run different diagnostics tests:

- *Environment Variables*
  Displays the Check Virtuoso & Unix Environment Setting for Best Performance form. This form provides options to report issues caused by:
  - Virtuoso and UNIX environment variables that have values different from the default values.
  - Virtuoso environment variables that are set to default values.

  You can also update Virtuoso environment variables as needed to fix an issue.
- *Cadence Generic DM*
  Displays the Test GDM form. In the form, provide the *Library*, *Cell*, and *View* information for a DM-managed cellview. The test will measure the real, user, and system time to get its current status. This test helps in checking the performance of your DM system.
- *Graphics Benchmark*
  Displays the Graphics Performance Benchmark form. Use this form to evaluate the ability of a platform to run the various graphics operations required by Virtuoso applications. For more information, see Measuring Graphics Performance.

- *System*
  Displays a text window that provides information regarding your system.



Running the SKILL Profiler

➡ To run the SKILL Profiler, in CIW, choose *Tools – Performance Diagnosis – SKILL – Profiler.*

The SKILL Profiler form is displayed.

You can use this form to retrieve SKILL callstack in two modes - time and memory.

The form has the following buttons on the left panel:

- *Setup*
  Displays the SKILL Setup Profiler form that lets you choose one of the two modes in which to retrieve the SKILL callstack:

- *Time spent in SKILL functions*
- *Memory allocated in SKILL functions*

- *Start* and *Stop*
  Starts retrieving SKILL callstacks when you click the *Start* button before running the slow command. Click the *Stop* button to stop the process.

  The results will display the total CPU time/allocated memory per SKILL function in a window.

- *Reset*
  Clears the profiling data to start a fresh measurement.

- *Browse*
  Displays the profiling data in a tree graph. Click a function name to expand the corresponding child node in this tree.

➡ Click the *File – Save As* command to save the data and send it Cadence support for further investigation.

Controlling the SKILL Reply

Interactively play a replay file from a `cdsmps` control console. Type `h` to see all commands, including setting a breakpoint to stop the replay or a starting line number. This is particularly useful to narrow down a reproducible test case from a large replay file.

➡ To start, in CIW choose *Tools – Performance Diagnosis – SKILL – Replay*. Then follow the instructions in CIW.

E

Using the oaScan Utilities for Virtuoso

oaScan is an unlicensed application that scans the contents of a library; checks for inconsistencies in the OpenAccess design, technology, and DMData databases; and optionally repairs the inconsistencies and saves the databases.

Inconsistencies are more common with older versions of OpenAccess, such as those released earlier than April 2008. oaScan can be used to upgrade inconsistent data to current quality standards and is maintained on an ongoing basis so that OpenAccess users can continuously monitor the integrity of their data.

Recommended Methodology

Cadence recommends that you

- Scan data that was created or modified by older versions of OpenAccess in order to identify data inconsistencies and then perform a repair to correct any inconsistencies found.
- Conduct periodic data scans in case older versions of OpenAccess are reintroduced into the design flow and repair any inconsistencies found.
- Run oaScan on data as it is being checked into a global workspace from a user's local workspace when using a design management system.

This appendix describes three GUI-based utilities that you can load in Virtuoso and access from the Command Interpreter Window (CIW) to help you implement the recommended methodology. For more information, see

- Scanning a Library to run oaScan on a library or cellview
- Scanning a Hierarchy to run oaScan on all the cellviews in a design hierarchy
- Scanning Cellviews Automatically during Save to run oaScan every time a cellview is saved

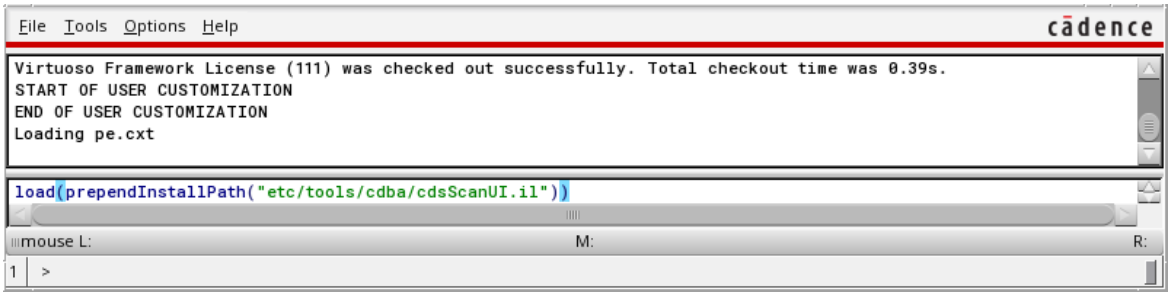
Full details of the functionality underpinning these utilities is available in the *OpenAccess oaScan User Guide*.

Enabling the oaScan Utilities

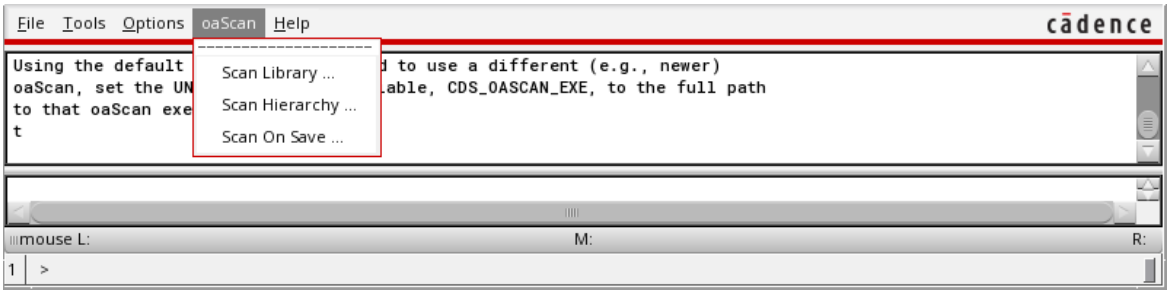
By default, the oaScan utilities are hidden in Virtuoso. To enable them:

➡ Type the following in the CIW:

```
load(prependInstallPath("etc/tools/cdba/cdsScanUI.il"))
```



The *oaScan* menu is added to the menu bar:



Specifying the oaScan Version to Use

By default, the utilities use the version of oaScan shipped with your Virtuoso hierarchy. To check the version installed, type the following in a terminal window:

```
%> oaScan -v
Tool:     oaScan     oaScan_p040 (22.50)
```

To use a different version of oaScan (for example, if you have received an updated version from Cadence), set the `CDS_OASCAN_EXE` shell environment variable to point to the oaScan binary you want to use:

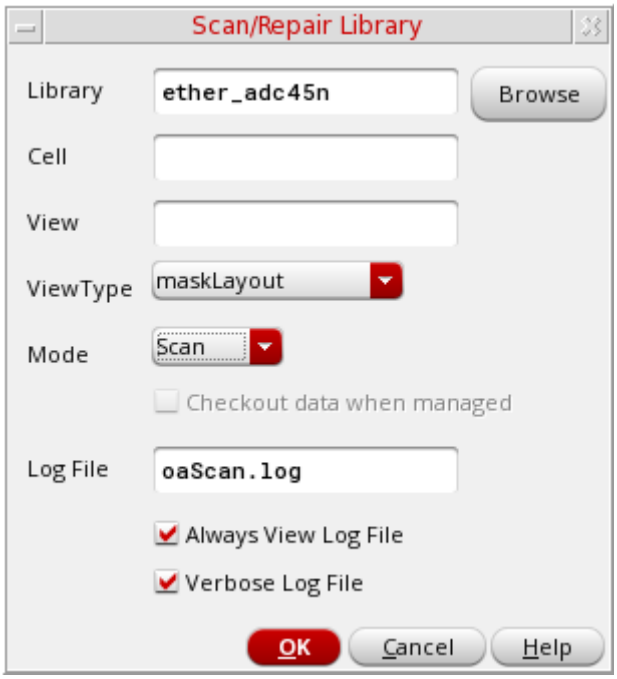
```
%> setenv CDS_OASCAN_EXE path/oaScan
```

Scanning a Library

To scan an entire library or a particular design database for OpenAccess issues:

1. From the CIW menu bar, choose *oaScan – Scan Library*.

   The Scan/Repair Library form is displayed.



2. Specify the name of the library to scan and, optionally, the cell and view names if you want to perform the scan on a particular design database.
3. Choose a *ViewType* to limit the scan to only views of the specified type.
4. Select the *Mode* in which the design is to be scanned:
   - *Scan* reports issues found but does not repair them.
   - *Repair* automatically repairs any issues found in the database. If your data is managed, you can also use the *Checkout data when managed* option to check out data from the repository prior to making any repair.
5. Specify a name for the log file (the default is `oaScan.log`).

   You can also specify that the log file is always displayed on completion and that it lists all designs processed rather than only those that have issues.
6. Click *OK* to scan the specified database.

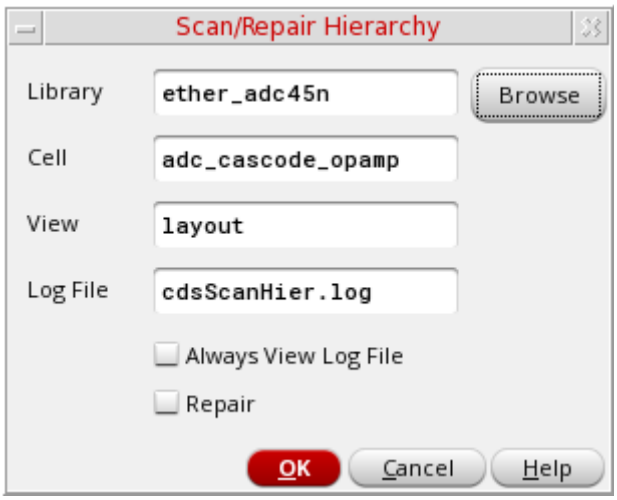
Scanning a Hierarchy

You can scan all the cellviews in a specific design hierarchy, starting from a specified top-level cellview. oaScan traverses the design hierarchy and generates a shell script containing one oaScan call for each cellview in the hierarchy.

To do this:

1. From the CIW menu bar, choose *oaScan – Scan Hierarchy*.

   The Scan/Repair Hierarchy form is displayed.



2. Specify the library, cell and view names of the top-level cellview.
3. Specify a name for the log file (the default is `cdsScanHier.log`).
4. Check the *Always View Log File* box to specify that the log file is always displayed on completion of the scan.
5. Check the *Repair* box to automatically repair any issues found in the design hierarchy.
6. Click *OK* to scan the hierarchy.

Scanning Cellviews Automatically during Save

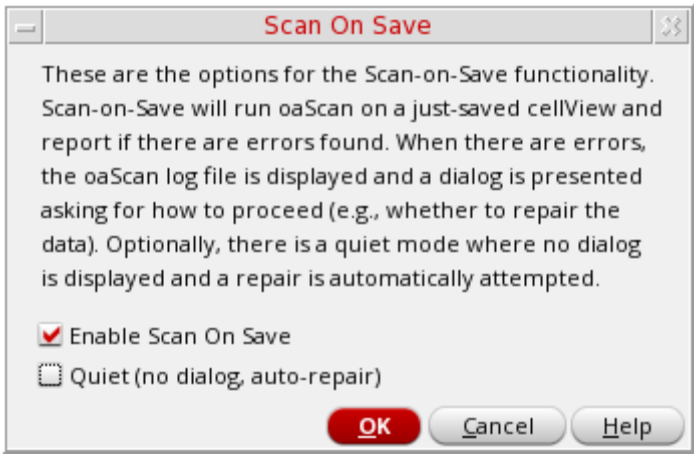
Instead of manually calling oaScan after each design update, you can run the check automatically any time you save data to disk. The software first makes a backup copy of the pre-save database and then runs oaScan on the newly-saved data. If a data issue has been introduced in the session since the last save, the backup copy can be restored.

Enabling Scan on Save

To enable Scan On Save:

1. From the CIW menu bar, choose *oaScan – Scan On Save*.

   The Scan On Save form is displayed.



2. Check the *Enable Scan On Save* box to switch on the feature.
3. Click *OK* to save the settings.

   Whenever you save a cellview, Virtuoso automatically makes a backup copy of the cellview and runs oaScan.

   - If the cellview is clean, a message to confirm this is issued in the CIW.
   - If the saved data has an issue, Virtuoso pops up a dialog where you can confirm what action to take next. See Handling Scan On Save Results for more information.

Specifying the Location of Scan On Save Log Files

By default, the log files generated by Scan On Save are saved to the `/tmp` directory.

Cadence recommends that you create a central location in which to store log files and point to it using the `CDS_SCANLOG` shell environment variable. For example:

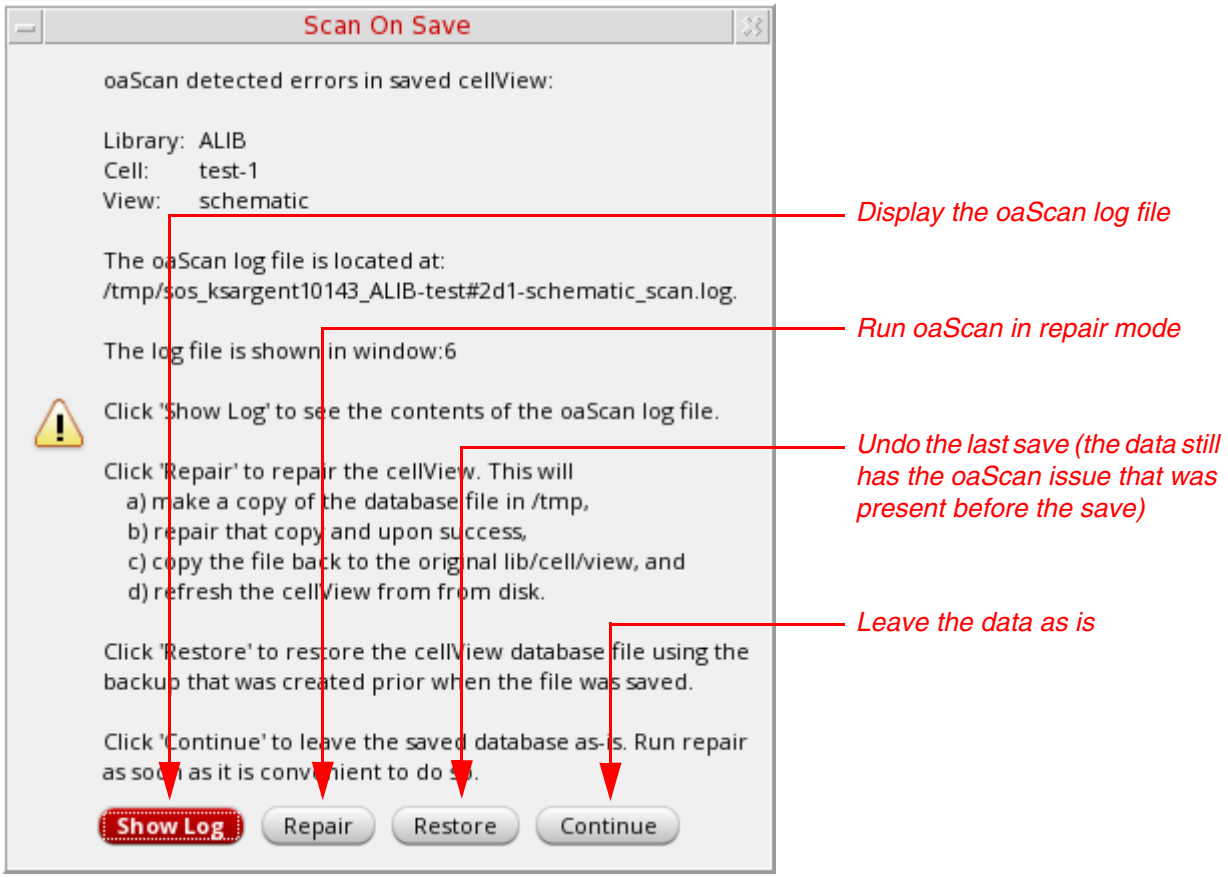
```
setenv CDS_SCANLOG "DIR=/home/user/scanlogs"
```

This lets you collect all oaScan log files in one place allowing you to easily locate and forward them to Cadence if required.

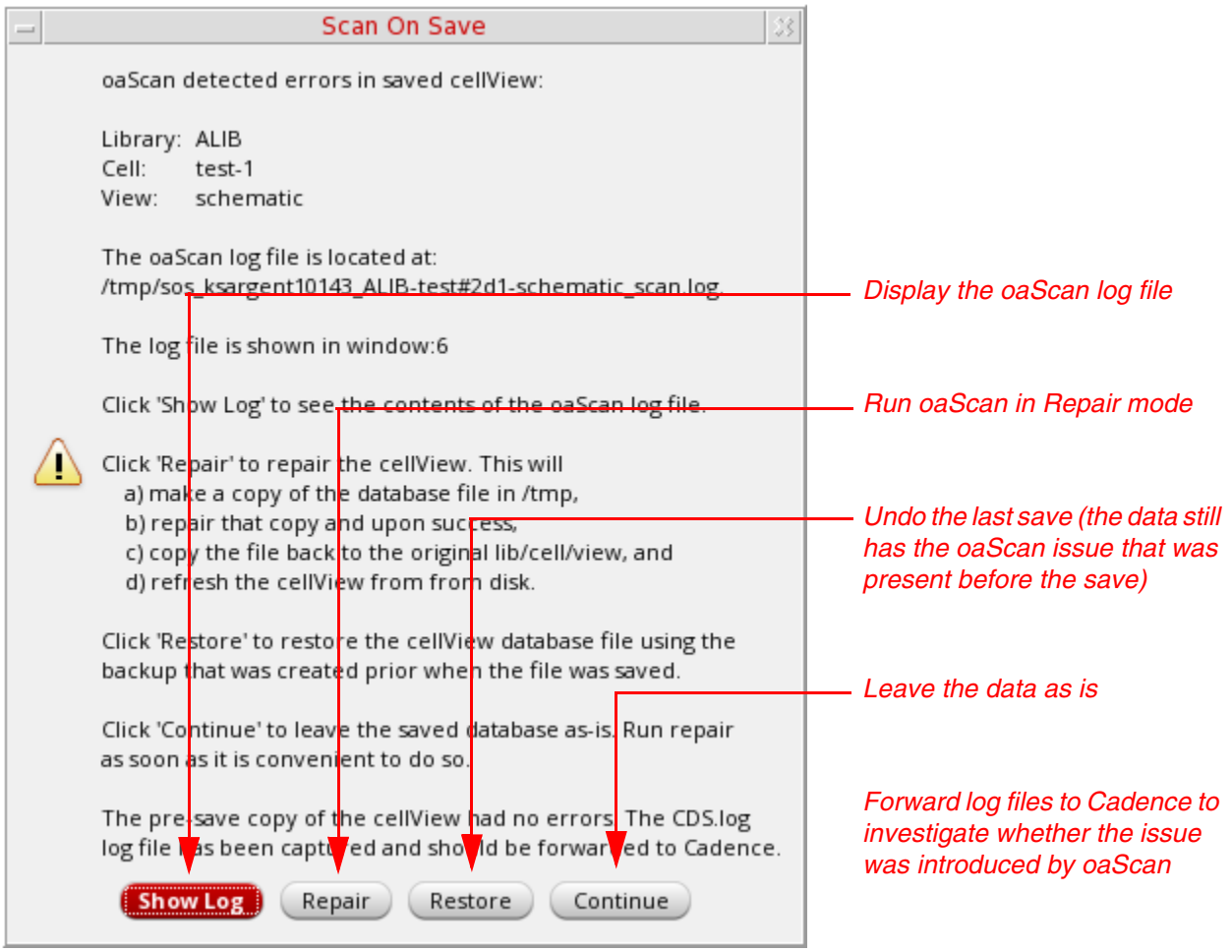
Handling Scan On Save Results

If the scan reveals an issue in the saved data, Virtuoso pops up a dialog where you can confirm what action to take next.

If the data contained an oaScan issue before the save, you see the dialog below:

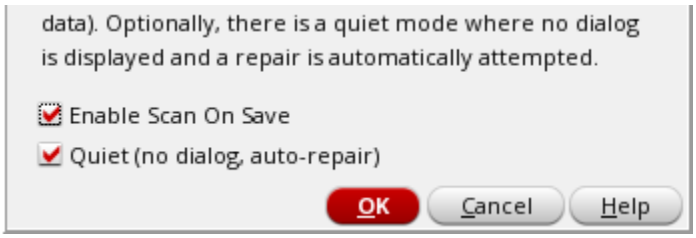


If an oaScan issue was found when the original data was clean, you see the dialog below:



Repairing Issues Automatically

To suppress the Scan On Save results dialogs and automatically repair any issues found, check the *Quiet* box in the Scan On Save form. When this setting is enabled, no dialog is popped up and oaScan attempts to automatically repair the cellview if required.



Forms

This section describes the forms available from the oaScan menu.

- Scan/Repair Hierarchy
- Scan/Repair Library
- Scan On Save

Scan/Repair Hierarchy

Use **Scan/Repair Hierarchy** to scan all the cellviews in a specific design hierarchy, starting from the specified top-level cellview. The software traverses the design hierarchy and generates a shell script containing one oaScan call for each cellview in the hierarchy.

Library, **Cell**, and **View** let you specify the top-level cellview of the hierarchy to be processed.

Log File specifies the name of the log file generated by oaScan. The default is `cdsScanHier.log`.

Always View Log File displays the log file on the desktop after each run.

Repair automatically repairs any issues found in the design hierarchy.

Related Topic

Scanning a Hierarchy

Scan/Repair Library

Use **Scan/Repair Library** to scan an entire library or a specific design database for OpenAccess issues.

Library specifies the library to be scanned.

Cell lets you specify the name of a cell to be processed. If not specified, all the cells in the library are processed.

View lets you specify a view name to be processed. If the *Cell* option is specified, oaScan processes the cellview in question. If the *Cell* option is not specified, all views with the specified name are processed.

ViewType limits the scan to databases of the selected type.

Mode lets you choose the mode in which the design is processed.

- *Scan* reports issues found but does not repair them
- *Repair* automatically repairs any issues found in the database

  Checkout data when managed attempts to automatically check out and repair managed data when required. (This option is available only if your data is managed.)

Log File specifies the name of log file generated by oaScan. The default is `oaScan.log`.

Always View Log File displays the log file on the desktop after each oaScan run.

Verbose Log File list in the log file information on all the designs processed rather than only those that have issues.

Related Topic

Scanning a Library

Scan On Save

Use the **Scan On Save** form to run oaScan automatically any time you save data to disk.

Enable Scan On Save switches on automatic checking during cellview save.

Quiet (no dialog, auto-repair) suppresses the pop up results dialog and attempts to automatically repair any issue found.

Related Topics

Specifying the Location of Scan On Save Log Files

Enabling Scan on Save

Handling Scan On Save Results

Repairing Issues Automatically

F

Glossary of Terms

assistant menu

Menu that appears at the top of an *assistant pane*. This menu differs from a *banner menu* which appears at the top of a *session window*.

assistant pane

Dockable *workspace* component that can have an *assistant toolbar* and an *assistant menu* bar. You can rearrange and resize assistant panes in your *session window*.

assistant toolbar

Toolbar that appears at the top of an *assistant pane*. This toolbar differs from a *banner toolbar* which appears near the top of a *session window*.

banner menu

Fixed menu that appears along the top of a *session window*. A banner menu differs from an *assistant menu* which can be found at the top of an *assistant pane* (when docked).

banner toolbar

Toolbar that appears at the top of the *session window* beneath the *banner menu*. A banner toolbar differs from an *assistant toolbar* which can be found at the top of an *assistant pane*.

bindkey

Keyboard key or mouse button linked (bound) to a Cadence SKILL command or function name.

CIW

Abbreviation for Command Interpreter Window.

session window

The current working window. A session window can contain multiple tabs, each potentially running a different application. You can have more than one session window open at a time, each with a *workspace* of its own.

task

Work objective. For example, a hierarchical task comprising a complex objective such as optimizing device sizing or a simple task such as adding a new pin to a schematic.

VNC

Abbreviation for Virtual Network Computing which allows you remote access to and control of an X Window System running Cadence software.

window controls

Settings that affect the position and appearance of windows in the Virtuoso user interface.

window element

Any configurable part of a window. For example, a *banner menu* or an *assistant pane*.

workspace

Selection of *window elements* comprising a particular layout of toolbars, and *assistant panes*. You can select and use Cadence workspaces or save and use customized workspaces