

Virtuoso Studio Design Environment User Guide

**Product Version IC23.1
November 2023**

© 2023 Cadence Design Systems, Inc.

Printed in the United States of America.

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

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor

Contents

1

<u>Virtuoso Studio Design Environment</u>	15
<u>Hardware and Software Requirements</u>	17
<u>System Configuration Checking Tool</u>	17
<u>Linux Requirements</u>	19
<u>Run Modes of Virtuoso Studio Design Environment</u>	21
<u>Interaction with the User Interface</u>	22
<u>Types of Menus</u>	22
<u>Functionality of Pointer and the Cursor</u>	23
<u>Functionality of Pop-Up Menus and Forms</u>	24
<u>Select and Deselect Items</u>	25
<u>Menu Access Keys</u>	26
<u>Setting Access Keys for Menu Items</u>	26
<u>Setting Access Keys for Menus</u>	27
<u>Escaping Ampersand</u>	27
<u>Guidelines for Creating Menu Access Keys</u>	27
<u>Menu Access Key Conflicts</u>	28
<u>Disable Menu Access Keys</u>	28
<u>Compatibility Issues</u>	28
<u>File Storage</u>	30

2

<u>Environment Settings in Virtuoso</u>	33
<u>Design Environment Customization</u>	33
<u>Virtuoso Studio Design Environment Files</u>	35
<u>User Preference Options</u>	40
<u>Log Filter Options</u>	42
<u>Viewing the Font List and Setting Fonts</u>	44
<u>GUI Default Setup Using .cdsinit</u>	45
<u>Opening Cdsenv Editor in CIW</u>	46
<u>Editing an Environment Variable in Cdsenv Editor</u>	47

Virtuoso Studio Design Environment User Guide

<u>License Checkout Behavior in .cdsenv</u>	49
<u>Use Next License Variables</u>	49
<u>Specifying New Default Values for a Virtuoso Session</u>	51
<u>Recalling Defaults</u>	52

3

<u>Virtuoso Studio Design Environment Launch</u>	53
<u>Starting Cadence Software</u>	54
<u>Command-Line Options For Cadence Applications</u>	55
<u>Starting Virtuoso Studio</u>	60
<u>Saving Changes in Virtuoso Studio Design Environment</u>	63
<u>Saving and Renaming Log Files in Virtuoso Studio Design Environment</u>	64
<u>Backing Up Your Work</u>	65
<u>Unlocking an Application</u>	66
<u>Exiting Virtuoso</u>	67

4

<u>Command Interpreter Window</u>	71
<u>File Menu in the CIW</u>	72
<u>Tools Menu in the CIW</u>	74
<u>Options Menu in the CIW</u>	76
<u>Tearing Off a Menu in the CIW</u>	77
<u>Moving and Resizing a Menu Window</u>	79
<u>Get Help with Virtuoso Menus</u>	80
<u>Session Commands and System Responses</u>	84
<u>Mouse Bindings Line</u>	85
<u>Actions Required on the Prompt Line</u>	86
<u>Mouse Controls in the Virtuoso Studio Design Environment</u>	87
<u>License Activity Indicator</u>	89
<u>Virtuoso Performance Warning Messages</u>	90
<u>Low Memory Warnings</u>	90
<u>Swap Activity Warnings</u>	91

5

<u>Working with Cellviews</u>	93
<u>Opening a Cellview from the CIW</u>	94
<u>Enable Design Preview</u>	96
<u>Cellview Already in Use</u>	97
<u>Indication of Modified Cellviews</u>	97
<u>Cellview History</u>	98
<u>Filtering Unavailable Tiers when Opening a Cellview</u>	98
<u>Setting the Default Application for a Cellview Type</u>	99
<u>Closing Cellviews in Virtuoso</u>	101
<u>Closing All Cellviews</u>	102
<u>Copying the Current Cellview to a New Session Window</u>	105
<u>Saving Modified Data</u>	106
<u>Default Instance Prefixes</u>	108
<u>Changing the Prefix Using a Property</u>	109
<u>Automatic Defragmentation of OpenAccess Databases</u>	111

6

<u>Virtuoso Studio Design Environment Customization</u>	113
<u>.cadence Hierarchy</u>	115
<u>Specifying User Preferences</u>	118
<u>Saving and Restoring Window Positions in Virtuoso</u>	120
<u>Restoring Window Positions of the CIW</u>	120
<u>Saving and Restoring the Position of the CIW Using SKILL</u>	121
<u>Changing the Log Filter Options</u>	122
<u>Menu Banner Customization</u>	124
<u>Customize Toolbar Definition Files</u>	125
<u>Toolbar Definition File Format</u>	125
<u>Toolbar Definition Search Path and Customization</u>	127
<u>Format for Main Toolbar Definition</u>	129
<u>Format for Item Type Action</u>	130
<u>Format for Item Type ComboBox</u>	133
<u>Format for Item Type Typein, Separator, and InheritToolbarsForm</u>	135
<u>Using and Resetting Toolbars</u>	137

Virtuoso Studio Design Environment User Guide

<u>Reset the Toolbar</u>	142
<u>Troubleshooting Information for Toolbar Manager</u>	143

7

<u>Virtuoso Workspaces</u>	145
<u>Workspace Configuration</u>	146
<u>Workspace Search Order</u>	148
<u>Session Windows and Workspaces</u>	151
<u>Tabs in a Session Window</u>	151
<u>Tab Pop-Up Menu Options</u>	152
<u>Hierarchy Changes and Workspaces</u>	154
<u>Workspace Features</u>	155
<u>Selecting a Workspace</u>	157
<u>Adding Assistant Panes and Toolbars to a Workspace</u>	158
<u>Removing Items from a Workspace</u>	161
<u>Controlling Assistant Panes</u>	163
<u>Displaying Assistant Panes as Tabs</u>	163
<u>Hiding Assistant Panes</u>	164
<u>Modifying the Size and Location of an Assistant Pane</u>	166
<u>Modifying the Size of a Docked or Floating Assistant Pane</u>	166
<u>Changing the Location of a Docked Assistant Pane</u>	166
<u>Changing the Location of a Floating Assistant Pane</u>	166
<u>Modifying the Length or Location of a Toolbar</u>	167
<u>Saving a Workspace</u>	168
<u>Loading and Deleting a Custom Workspace</u>	170
<u>Deleting a Custom Workspace</u>	170
<u>Customizing a Cadence Workspace</u>	172
<u>Reverting to a Saved Workspace</u>	172
<u>Setting the Default Workspace for an Application</u>	173

8

<u>Design Editor Plugins</u>	175
<u>Features of a Design Editor Plugin</u>	175
<u>Defining and Customizing New Toolbars</u>	176
<u>Support Bindkeys in DE Plugin</u>	176

Virtuoso Studio Design Environment User Guide

<u>Banner Menus</u>	176
<u>Workspace Defined by DE Plugin</u>	177
<u>Accessing a Design Editor Plugin</u>	178
<u>Registering a Design Editor Plugin</u>	180
<u>Installing and Removing a Design Editor Plugin</u>	181
<u>Converting an Existing Application to a Plugin</u>	183

9

Bookmarks and Views in Virtuoso Studio Design Environment

185

<u>Bookmarking Designs</u>	187
<u>Restoring a Bookmark</u>	189
<u>Restoring a Single Bookmark</u>	189
<u>Restoring a Composite Bookmark</u>	190
<u>Restoring a View from a Composite Bookmark</u>	191
<u>Managing Bookmarks</u>	193
<u>Adding a Bookmark</u>	194
<u>Importing Bookmarks</u>	194
<u>Exporting Bookmarks</u>	195
<u>Reordering Bookmarks</u>	195
<u>Deleting Bookmarks</u>	196
<u>Editing Bookmark Properties</u>	196
<u>Opening Bookmarked Views</u>	197
<u>Searching for Bookmarks</u>	198
<u>Managing Composite Bookmarks</u>	199
<u>Adding Bookmarks to Existing Composite Bookmarks</u>	199
<u>Creating Composite Bookmarks from Existing Bookmarks</u>	200
<u>Detaching Bookmarks from Existing Composite Bookmarks</u>	201
<u>Saving and Restoring Views</u>	203
<u>Restoring a Saved View</u>	203
<u>Restoring the Previous View</u>	204
<u>Restoring the Next View</u>	204

10

Navigating Cellviews and Hierarchies 205

Accessing the Go Toolbar 207

Moving Back through a Design 208

Moving Forward through a Design 210

Moving Up and to the Top of a Design 212

11

Text Window Options 213

Saving a Text File 214

Searching a File for Specific Text 215

Printing and Refreshing a File 217

12

Design Object Limits in OpenAccess 219

13

Importing and Exporting Designs 221

14

Design Data Scaling 223

Changing the DBUperUU in the Technology File 224

XScale Command 225

A

Bindkeys and Access Keys 227

Viewing the Current Bindkeys for an Application 229

Restrictions to Setting Bindkeys 231

Configuring Application Bindkeys 232

Adding a Bindkey 232

Capturing Mouse and Key Bindings 233

Duplicate Bindkeys 233

Virtuoso Studio Design Environment User Guide

<u>Deleting a Bindkey</u>	234
<u>Editing a Bindkey</u>	234
<u>Format for Key and Mouse Bindings</u>	236
<u>Guidelines for Non-ASCII-7 Character Bindkey Definitions</u>	239
<u>Determining SKILL Function for a Menu Command</u>	241
<u>Setting Bindkeys Across Sessions Using the .cdsinit File</u>	242
<u>Control Bindkey Display on Menus</u>	243
<u>Default Keybindings for Text Fields</u>	245
<u>Default Keybindings for CIW</u>	246
<u>Default Keybindings for Forms</u>	251
<u>Rules for Bindkey Use with Keypads</u>	252
<u>Access Keys</u>	253

B

<u>Design Environment Variables</u>	255
<u>CDBA Environment Variables</u>	256
<u>AlternateFoundryCG</u>	257
<u>copyMPAttributes</u>	258
<u>dbAddCellNameToInstNamePrefix</u>	259
<u>dbAllowStdViaCutLayerOverride</u>	260
<u>dbArrayInstNamePrefix</u>	261
<u>dbEnableRouteObservers</u>	262
<u>dbFigGroupNamePrefix</u>	263
<u>dbInstNamePrefix</u>	264
<u>dbLogPcellWarnings</u>	265
<u>dbNumCPU</u>	266
<u>dbUndoAcrossPurge</u>	267
<u>dbUndoAcrossSave</u>	268
<u>dbUpdateCellNameInInstNamePrefixDuringRemaster</u>	269
<u>defaultAttachTech</u>	270
<u>disablePartialRead</u>	271
<u>maxMasterSize</u>	272
<u>noDetailedRowCol</u>	273
<u>noTechUpRev</u>	277
<u>propsToAppend</u>	278

Virtuoso Studio Design Environment User Guide

<u>sessionUsageType</u>	280
<u>resetLibList</u>	281
<u>resetOnRemaster</u>	282
<u>usePerDesignAFCCG</u>	283
<u>verbosity</u>	284
<u>closeDataOption</u>	285
<u>closeDataSaveOption</u>	286
<u>importDuplicateBookmarks</u>	287
<u>numThreads</u>	288
<u>addVLSEXLUserTriggersToMXL</u>	289
<u>addVLSXLUserTriggersToMXL</u>	290
<u>resetUndoOnDescending</u>	291
<u>ignoreAppTierInHistory</u>	292
<u>cellviewModifiedIndicator</u>	293
<u>fitViewAttempts</u>	294
<u>iconNameFormat</u>	295
<u>windowNameFormat</u>	296
<u>Performance Environment Variables</u>	298
<u>autoLogLatency</u>	300
<u>cpuUtilization</u>	301
<u>deleteHMLLog</u>	302
<u>enableExpertMode</u>	303
<u>fileTag</u>	304
<u>gdbPath</u>	305
<u>installAtStartup</u>	306
<u>installPath</u>	307
<u>launchUI</u>	308
<u>logDir</u>	309
<u>memUtilization</u>	310
<u>openReportForm</u>	311
<u>pinnedOffsetX</u>	312
<u>popUpLog</u>	313
<u>postReportScript</u>	314
<u>preReportScript</u>	315
<u>privateVarAtStartup</u>	316
<u>pstackPath</u>	317

Virtuoso Studio Design Environment User Guide

<u>stracePath</u>	318
<u>sysMonitorPath</u>	319
<u>sysPulseRefreshInterval</u>	320
<u>sysPulseYellowLightAlert</u>	321
<u>toolbarAutohide</u>	322
<u>toolbarBorderless</u>	323
<u>toolbarNativeWM</u>	324
<u>virtuosoPulseRefreshInterval</u>	325
<u>virtuosoPulseYellowLightAlert</u>	326
<u>User Interface Environment Variables</u>	327
<u>accelInput</u>	328
<u>accelReturnValue</u>	329
<u>beepVolume</u>	330
<u>bottomAsstSpansFullWidth</u>	331
<u>ciwCmdExecuteOnEnter</u>	332
<u>ciwCmdHistoryInPlace</u>	333
<u>ciwCmdInputLines</u>	334
<u>ciwErrorColor</u>	335
<u>ciwLogHistorySize</u>	336
<u>ciwMatchCmdColor</u>	337
<u>ciwMatchParenColor</u>	338
<u>ciwMismatchParenColor</u>	339
<u>ciwOutputWrapMode</u>	340
<u>ciwRetainUniqueCmds</u>	341
<u>ciwSyntaxHighlighting</u>	342
<u>ciwTabStop</u>	343
<u>ciwWarnColor</u>	344
<u>dblClkTime</u>	345
<u>defaultDragColor</u>	346
<u>defaultEditorBackgroundColor</u>	347
<u>defaultFloatFieldFormat</u>	348
<u>enableFileDialogNameCompletion</u>	349
<u>errorOutput</u>	350
<u>floatPrecision</u>	351
<u>focusToFieldSelectsText</u>	352
<u>formDefaultAction</u>	353

Virtuoso Studio Design Environment User Guide

<u>imageTabTip</u>	354
<u>interruptCheckInterval</u>	355
<u>maximumCopySize</u>	356
<u>memoryCheckintervalSeconds</u>	357
<u>mouseMoveSampleRate</u>	358
<u>mouseStopDetectTime</u>	359
<u>mouseWheelSpeed</u>	360
<u>nestLimit</u>	361
<u>noWarnOnLVBindKeyCustomization</u>	362
<u>openOptionsFormAtMousePos</u>	363
<u>optionFormsStayOnTop</u>	364
<u>printAllLVBindKeyBlockWarning</u>	365
<u>promptOutput</u>	366
<u>raiseCIWonError</u>	367
<u>raiseCIWonWarning</u>	368
<u>releaseBackingStoreOnUnmap</u>	369
<u>rememberOptionFormVisibility</u>	370
<u>showMouseBar</u>	371
<u>showOptionForms</u>	372
<u>sideDockTabs</u>	373
<u>stopLevel</u>	374
<u>systemMemoryCheckinterval</u>	375
<u>typedReturnValue</u>	376
<u>undoLevel</u>	377
<u>useLineStyleForStrokeText</u>	378
<u>warningOutput</u>	379
<u>standardOutput</u>	380
<u>webBrowser</u>	381

C

<u>Virtuoso Studio Design Environment Forms</u>	383
<u>Add Bookmark Form</u>	385
<u>Auto Checkin Preferences Form</u>	386
<u>Auto Checkout Preferences Form</u>	387
<u>Benchmark Results Window Form</u>	388

Virtuoso Studio Design Environment User Guide

<u>Bindkey Editor Form</u>	389
<u>Cdsenv Editor Form</u>	393
<u>Design Form</u>	395
<u>Environment Form</u>	398
<u>Library Form</u>	400
<u>SKILL Form</u>	401
<u>Check Environment Variables Form</u>	402
<u>Close and Purge Data Form</u>	404
<u>Close Opened Cellviews Form</u>	405
<u>Conversion Toolbox Form</u>	406
<u>Convert DE-HDL libraries to Unified libraries</u>	407
<u>Data Storage Speed Test Form</u>	408
<u>Diagnostic Center Form</u>	409
<u>Performance</u>	409
<u>Data Integrity</u>	410
<u>Edit Bookmark Properties Form</u>	411
<u>Edit Library Path Form</u>	412
<u>Email Tabulated List of Functions Form</u>	413
<u>File Preferences Form</u>	414
<u>Graphics Performance Benchmarks Window Form</u>	416
<u>Health Monitor Form</u>	419
<u>Health Monitor (Advanced) Form</u>	423
<u>Main</u>	423
<u>Log</u>	424
<u>System</u>	425
<u>Options</u>	427
<u>Load Workspace Form</u>	429
<u>Make Read Only Form</u>	430
<u>Monitor by Strace Form</u>	431
<u>New Hierarchy Form</u>	432
<u>New Library Form (CIW)</u>	433
<u>Open File Form</u>	434
<u>Save .cdsenv file Form</u>	435
<u>Save As Form</u>	437
<u>Save Cellviews Form</u>	438
<u>Save Defaults Form</u>	439

Virtuoso Studio Design Environment User Guide

<u>Save Modified Data Form</u>	441
<u>Save Session Form</u>	442
<u>Save View Form</u>	443
<u>Scan/Repair Hierarchy Form</u>	444
<u>Scan/Repair Library Form</u>	445
<u>Scan On Save Form</u>	446
<u>Search Form</u>	447
<u>Set Fonts Form</u>	448
<u>Set Log File Display Filter Form</u>	449
<u>Show File Form</u>	450
<u>SKILL Name Checker</u>	451
<u>Surveyor: SKILL Tabulator Form</u>	452
<u>Software Product License Management Form</u>	454
<u>Unable to check out Form</u>	455
<u>User Preferences Form</u>	456
<u>ViewFile Window Form</u>	463
<u>What's New Search Form</u>	466

D

<u>Virtuoso SKILL Interface</u>	467
<u>Running SKILL Commands</u>	467
<u>SKILL Commands Execution</u>	468
<u>Debug Support for SKILL Commands</u>	469
<u>Command Repetition in CIW</u>	471
<u>Setting the SKILL Search Path and Specifying the Editor</u>	473
<u>Operating SKILL Surveyor</u>	474
<u>Specifying Alternative Mail Command Path</u>	475
<u>Troubleshooting SKILL Issues in Virtuoso</u>	475

E

<u>Diagnostics</u>	477
<u>Application Crash Reporting</u>	478
<u>Crash Report Customization</u>	481
<u>Crash Trend Reporting</u>	487
<u>Availability of Signal Description Information for Crashes</u>	488

Virtuoso Studio Design Environment User Guide

<u>Storing User Feedback on Crashes</u>	488
<u>Crash Due to X Server Error</u>	488
<u>Crash Detector Display Check</u>	488
<u>Standard Virtuoso Exit</u>	489
<u>Additional Crash Data Collection</u>	490
<u>Crash Report Data Storage</u>	491
<u>Measuring Graphics Performance</u>	493
<u>Running Virtuoso Graphics Performance Benchmarks</u>	493
<u>hiGraphicsBenchmark Command-Line Arguments</u>	495
<u>Performing Benchmark Tests</u>	496
<u>Task Viewer</u>	499
<u>Performance Benchmarks</u>	501
<u>Diagnostic Center Overview</u>	505
<u>Health Monitor Overview</u>	507
<u>Collecting Data Using Health Monitor Tool</u>	510
<u>Reporting Performance Issues</u>	511
<u>Reporting Slowness</u>	512
<u>Troubleshooting an Unresponsive Virtuoso Application</u>	513
<u>Checking the Profiler Summary</u>	515
<u>Controlling the SKILL Replay</u>	515
<u>Automatic Diagnostic Log Submissions in Virtuoso</u>	517
<u>VAILES Enabled Script</u>	519
<u>Anonymization of Diagnostic Logs</u>	522
<u>Use of UNIX 'sed' Command in Anonymization</u>	524
<u>Conditions for Anonymization</u>	526
<u>Validation of Automatic Diagnostic Log Submission</u>	527
<u>vails_test Script to Launch Virtuoso using Job Scheduler</u>	532
<u>How to Send the Diagnostic Logs to Cadence</u>	533

F

<u>oaScan Utilities for Virtuoso</u>	535
<u>Recommended Methodology</u>	535
<u>Enabling the oaScan Utilities and Specifying the oaScan Version to Use</u>	537
<u>Specifying the oaScan Version to Use</u>	537
<u>Scanning a Library</u>	539

Virtuoso Studio Design Environment User Guide

<u>Scanning a Hierarchy</u>	541
<u>Scanning Cellviews Automatically During Save</u>	543
<u>Enabling Scan on Save</u>	543
<u>Specifying the Location of Scan On Save Log Files</u>	543
<u>Handling Scan On Save Results</u>	544
<u>Repairing Issues Automatically</u>	545

G

<u>Glossary of Terms</u>	547
--------------------------------	-----

Virtuoso Studio Design Environment

The main elements that the Virtuoso Studio Design Environment comprises are:

- **Workbenches:** These are binary files that contain blocks of functional computer code. Each workbench contains the code to run several related applications. You can start a workbench by typing 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.

Note: 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.

- **Command Interpreter Window (CIW):** It appears when you start the Virtuoso Studio Design Environment. You can use the CIW to access the Cadence application that you are licensed to run.
- **Common Desktop Environment (CDE):** It controls the size, placement, and behavior of windows.
- **Tabbed window architecture:** It is used for simultaneous editing of several cells and views.
- **Dockable assistant panes:** It is used for access to property editing and design-related tasks.
- **Improved hierarchical navigation**
- **Smart interactive search technology**

You can adjust various settings to customize your windows environment:

- To customize the settings that only impact Cadence software, you can use the associated menu selections or edit the `.cdsinit` and `.cdsenv` files.

For more information, see [Virtuoso Studio Design Environment Customization](#).

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment

- To customize X Window System settings, which control the look, feel, and behavior of both Cadence windows and other windows on your screen, you can edit the `.Xdefaults`, `.xsession`, and `.xinitrc` files.

Related Topics

[Virtuoso Studio Design Environment Launch](#)

[Session Windows and Workspaces](#)

[Adding Assistant Panes and Toolbars to a Workspace](#)

[Working with Cellviews](#)

[Command Interpreter Window](#)

[Virtuoso Workspaces](#)

[Navigating Cellviews and Hierarchies](#)

Hardware and Software Requirements

This section provides hardware, software, and file system requirements. The workstation on which you install the latest version 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.

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 are 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 gets significantly reduced. For higher resolutions, the Video Memory should be increased.

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 Xrender 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
```

XResource 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. When this happens, a warning is added to the `CDS.log` file. Additionally, during runtime, if there is an error in X server, Virtuoso checks and reports the X resources consumed by all clients running at that moment on Xserver. Information regarding the top five users of X client is also added to the `CDS.log` file. This information helps check if the error is related to X resource.

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. Virtuoso can tolerate up to 150ms of latency, but Cadence recommends that it stays lower.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment

Linux Requirements

These are the hardware requirements for Linux desktop:

Component	Minimum Requirement	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

These are the hardware requirements for Linux server:

Component	Minimum Requirement	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

These are the software requirements for Linux:

Component	Requirement
Operating System	Red Hat Enterprise Linux (RHEL) 7; 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

Updating Patches for Linux

Cadence recommends that you update your workstations to the platform patches listed at:

Recommended platform patches for systems running Cadence products

Alternatively, go through Cadence Online Support (COS):

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*.

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

Related Topics

Cadence Online Support

System Configuration Checker (checkSysConf)

Run Modes of Virtuoso Studio Design Environment

When you start the Virtuoso Studio Design Environment in graphics mode, the Command Interpreter Window (CIW) appears. 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. For more information, see [Starting Cadence Software](#).

When Virtuoso has launched, the time taken to successfully checkout the 111 (Virtuoso Framework) license is 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.

Related Topics

[Command Interpreter Window](#)

Interaction with the User Interface

This section lists down all the useful interactions that you may encounter using menus in Virtuoso Studio Design Environment.

Types of Menus

A menu is a list of related commands. Virtuoso Studio Design Environment uses the following kinds of menus:

- Pull-down menus: These menus are menus that are attached to the menu bar and that are displayed when you click a menu title to pull down the menu.

You create a pulldown menu with `hiCreatePulldownMenu` using previously created menu items. After you create a pulldown menu, attach it to the menu bar of a window.

Menu items in pulldown menus can be text, icon, or slider items. They can be enabled or disabled. They can also have callback functions associated with them.

Some text menu items in pulldown menus, such as *Load*, *Open*, *Save As*, *Undo*, *Redo*, *Close*, *Exit*, and *Print*, automatically display an icon next to the text. You can choose not to display the default icons for these menu items by setting the following variable in your `.cdsenv` file:


```
ui useAutoPixmaps boolean nil
```

- Pop-up menus: These menus that are not visible until the user displays them, typically with a right-click on an application window. Popup menus contain context-sensitive choices.

You can create popup menus with the following SKILL functions:

- ❑ `hiCreateSimpleMenu`
Creates a menu with text menu items only. You can create this menu and its items at the same time, without having to create menu items ahead of time.
- ❑ `hiCreateMenu`
Creates a menu with text, icon, or slider menu items. Menu items are laid out in a column, top to bottom, in a specified order. Items can also be disabled (greyed-out) so that they are unselectable.
- ❑ `hiCreate2DMenu`
Creates a menu with text or icon menu items, or inactive labels. This menu type lets you specify the exact placement of each menu item through an attribute list provided with each item.

Some text menu items in popup menus, such as *Load*, *Open*, *Save As*, *Undo*, *Redo*, *Close*, *Exit*, and *Print*, automatically display an icon next to the text. You can choose not to display the default icons for these menu items by setting the following variable in your

 `.cdsenv` file:

```
ui useAutoPixmaps boolean nil
```

You can set a `warpCursorBack` property on a popup menu that will store the cursor location at the time a popup menu is displayed.

For example, you can set the following menu property:

```
MyPopupMenu->warpCursorBack = t
```

Therefore, when set to `t` the cursor will be restored back to its original position, prior to the popup menu opening, when that popup menu is subsequently closed.

- **Toolbar icons:** Get displayed 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.

Functionality of 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:

Shape	Description
	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.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment

Shape	Description
	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.

Functionality of 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:

Action	Procedure
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. 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.

Select and Deselect Items

To select and deselect items:

Action	Procedure
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	Press <code>Shift</code> on the new end point.
Delete one item from a selection	Press the <code>Control</code> key and click the item.
Delete several items from a selection	Press <code>Control</code> and drag the pointer over the items to delete. When you are done, click <i>OK</i> .
Close the box without selecting an item	Click <i>Cancel</i> .

Related Topics

[User Preferences Form](#)

[Save Defaults Form](#)

[Run Modes of Virtuoso Studio Design Environment](#)

Menu Access Keys

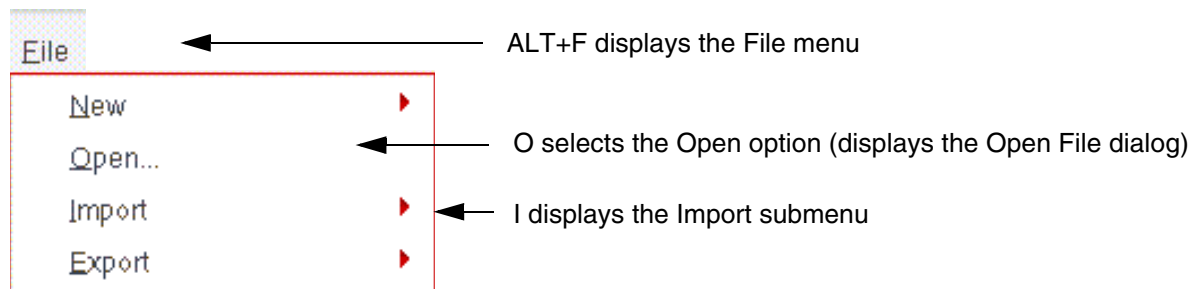
Menu access keys, also known as menu shortcuts, provide keyboard access to menus. With these keys, you can:

- Display a menu
- Select menu items from a displayed menu

without using the mouse. These keys provide a quick way of navigating and learning menus.

Menu access keys use the modifier `ALT`. To display a menu, you press `ALT` and the access key for that menu. To select an option in that menu, you then press the access key for that option (without `ALT`). For example, if the menu access key for the `File` menu is `F`, you would press `ALT+F` to display the menu. To select the `New` option in the menu, you would then press `N` (assuming that its access key is `N`.)

Menu access keys are displayed as underlined letters in the menu or menu item names. For example:



Menu access keys always display a menu, unlike bindkeys that do not display a menu or require a menu to be displayed in order to select menu items. For example, with a bindkey, you can select the `File • -New` option without the `File` menu being displayed.

Setting Access Keys for Menu Items

To set an access key for a menu item when you create the menu item, in the item text, specify `&` (ampersand) before the character that you want to designate as the access key. Specifically:

When you create a menu item with `hiCreateMenuItem`, add `&` before the desired character in the `?itemText` argument. For example, creates the menu item `New`, whose access key is `N`:

```
hiCreateMenuItem(?name 'new ?itemText "&New")
```

- ❑ When you create a menu item with `hiCreateSliderMenuItem`, add `&` before the desired character in the `?itemText` argument. For example, creates the menu item `Open`, whose access key is `O`:

```
hiCreateSliderMenuItem(?name 'open ?itemText "&Open" ?subMenu openOptions)
```

- ❑ When you create an action for a menu, specify `&` before the desired character in the `?iconText` argument of `hiCreateAction`. For example, creates the action `Refresh`, whose access key is `F`:

```
hiCreateAction(?name 'refresh ?iconText "Re&fresh" ?callback "refreshCB")
```

- ❑ When you create a simple menu with `hiCreateSimpleMenu`, specify `&` before the desired character in each menu item in the `l_menuItems` argument. For example, creates a menu with the menu items `Place` and `Print`, whose access keys are `P` and `R` respectively:

```
hiCreateSimpleMenu('customized "Custom" '("&Place" "P&rint") '(CB1 CB2))
```

Since symbols are created from the item text in this case, the symbol names also include the ampersands.

Setting Access Keys for Menus

To set an access key for a menu,

- † When you create the menu with `hiCreatePulldownMenu`, `hiCreateMenu`, `hiCreateSimpleMenu`, or `hiCreate2DMenu`, specify `&` (ampersand) in the menu title before the character that you want to designate as the access key.

For example, creates the `File` menu. Pressing `ALT+F` displays this menu:

```
hiCreatePulldownMenu('File "&File" list( new open close))
```

Escaping Ampersand

If you have an ampersand as part of a menu or menu item name, you need to escape it with another ampersand. For example, to display the name `P&R`, you need to specify the menu title or menu item text as `"P&&R"`. Otherwise, it will be displayed as `PR` and `R` will be the access key for it.

Guidelines for Creating Menu Access Keys

- Access keys always use the modifier `ALT`.
- Access keys can be any printable ASCII character, although they are typically alphanumeric characters.

- Access keys for menus must be unique within an application.

Many applications can launch other applications by adding their menus to the menu bar. Access keys for menu items must be unique within the menu or submenu.

- If a window manager or desktop, such as CDE, predefines an `ALT+key` binding, then that `ALT+key` combination will not work as a menu access key or bindkey on that system.

Menu Access Key Conflicts

You can display warnings about menu items that have conflicting access keys by setting the following variable in your `.cdsenv` file:

```
ui checkAccessKeyConflicts boolean t
```

Warnings are displayed in the CIW. This variable is set to `nil` by default.

Disable Menu Access Keys

Menu access keys can be disabled in one of the following ways:

- By adding the following line to your `.cdsenv` file:

```
ui enableMenuShortcuts boolean nil
```

- By deselecting the *Menu Shortcuts* option in the User Preferences form (displayed by selecting *Options – User Preferences* in the CIW) and then saving your defaults in the Save Defaults form (displayed by selecting *Options – Save Defaults* in the CIW).

Your change will take effect the next time that you start the software.

When you disable menu access keys, bindkeys can use `ALT`.

Compatibility Issues

- In your custom code, if you access Cadence menu or menu item names and do string comparisons against them, you need to be aware that there are now extra ampersands in the names because of access keys. You need to modify your code accordingly.

The code that accesses Cadence menus through menu handles or symbols is unaffected.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment

- If you have bindkeys that use `ALT`, you should redefine them because `ALT` is the modifier used with menu access keys. It can no longer be used with any printable characters in bindkeys, unless menu access keys are disabled.

Related Topics

[Run Modes of Virtuoso Studio Design Environment](#)

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. The Cadence software creates and edits the files in the `.cadence` hierarchy. Avoid editing the contents of `.cadence`. Cadence also installs a signature file to verify that Cadence has created the directory.

For more information on the `.cadence` hierarchy, see [.cadence Hierarchy](#).

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

You can choose to save your custom workspace and toolbar files 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:

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

Related Topics

[Cadence Setup Search File: setup.loc](#)

[Customizing a Cadence Workspace](#)

[Customize Toolbar Definition Files](#)

[Bookmarking Designs](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment

Environment Settings in Virtuoso

Environment settings and initialization instructions for Cadence applications are specified in `.cdsenv` and `.cdsinit` files, respectively. You can also use these files to customize your design environment settings. Your system administrator copies a customized version of `.cdsenv` and `.cdsinit` files to your local site:

`your_install_dir/tools/dfII/local`, there is an option to maintain local copies in your home directory. This file is present only if your site administrator copies a customized environment file to this location.

Before customizing these files you must customize your home directory. A home directory is specified by a tilde, such as `~/ .cshrc`. To make UNIX-level changes affecting file locations, paths, and display devices, you need to edit your `.cshrc` file, `.login` file or `.profile` file in your home directory. When the Cadence software loads, it reads the site-specific settings first and the individual settings last, so settings in your home directory override system and group settings.

Design Environment Customization

You can customize the following files in the Cadence software:

■ Log File

To specify a unique log file name using the `CDS_LOG_VERSION` environment variable, set the variable before starting the Cadence software or include the `setenv` command in your `.cshrc` file:

```
setenv CDS_LOG_VERSION {sequential|pid}
```

where,

- ❑ `sequential` adds a sequential number to the name of the log file, such as `CDS.log.1` or `CDS.log.2`
- ❑ `pid` adds the number of the UNIX process to the name of the log file, such as `CDS.log.1719` or `CDS.log.2250`

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

You can also set these additional variables in your `.cshrc` file.

Variable	Description
CDS_COLOR_MODE	Sets monochrome display of Cadence software. Value: <code>BlackAndWhite</code>
DISPLAY	Sets the visual display of the X Window System
FMHOME	Sets the path to FrameMaker product directory if not in the default directory

■ Sample Files

The Cadence software includes commented sample files in `install_dir/tools/dfII/samples`. The `install_dir` directory is the directory where the Cadence software is installed. You can also customize parts of your environment using the *Options* menu in the Command Interpreter Window.

Each file has a different format:

- ❑ The `.cdsenv` file contains settings of the form
`tool[.section] setting type value`
- ❑ The `.cdsinit` file contains executable commands for custom software initialization such as bindkey definitions, custom SKILL procedures, and user preferences.

■ Help Files

To specify the location of your Doc Assistant files, add the following command to your `.cshrc` file, `.login` file, or `.profile` file:

```
setenv HELPDIR install_dir/tools/dfII/myHelp
```

where `myHelp` is the name of the Help directory you want.

■ Search Path

To add the installation directory to your UNIX search path, use one of the following commands:

```
set path = ($PATH install_dir/tools/dfII/bin)
setenv PATH $PATH:install_dir/tools/dfII/bin
```

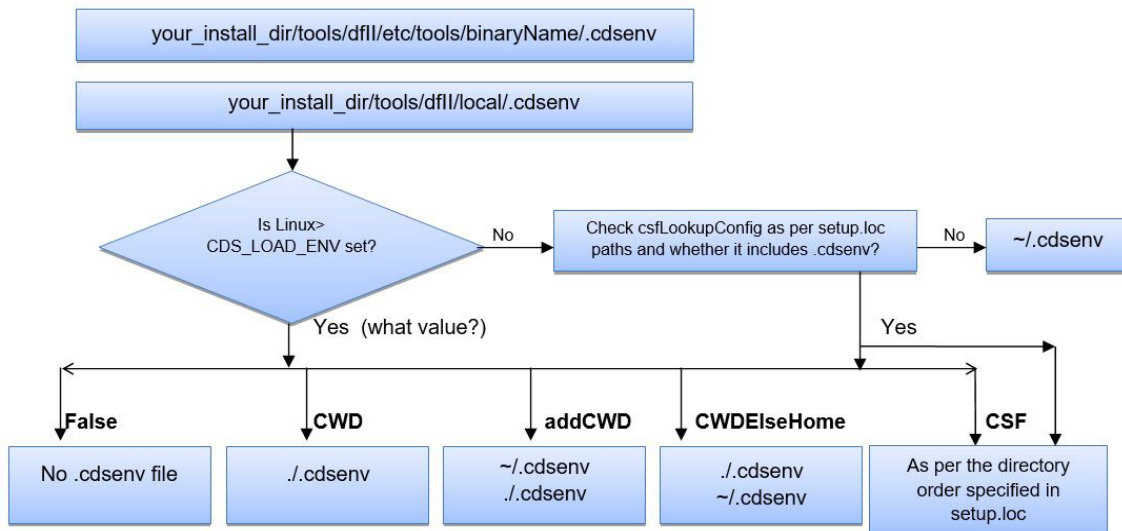
Additionally, you can specify `.cdsenv` and `.cdsinit` in your `csfLookupConfig` file, Virtuoso uses the Cadence Setup Search File mechanism to find this file.

Virtuoso Studio Design Environment Files

Virtuoso loads settings in the following order:

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

Flow chart .cdsenv file



- ❑ 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 appear in the output area.

- ❑ You can specify a custom search order for `.cdsenv` by setting the `CDS_LOAD_ENV` environment variable to one of the following values:

False	Load neither <code>~/cdsenv</code> nor <code>CWD/.cdsenv</code>
CWD	Load <i>currentWorkingDirectory/.cdsenv</i> instead of <code>~/cdsenv</code>
addCWD	Load <code>~/cdsenv</code> , then load <i>currentWorkingDirectory/.cdsenv</i>
CWDElseHome	Load <i>currentWorkingDirectory/.cdsenv</i> if it exists, otherwise <code>~/cdsenv</code> , if it exists

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

CSF	Load <code>.cdsenv</code> files according to the Cadence Setup Search File mechanism
	Specifying <code>.cdsenv</code> in your <code>csfLookupConfig</code> file is equivalent to setting <code>CDS_LOAD_ENV</code> to CSF.
	Environment variable: <code>csfLookupConfig</code>

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.
- The `CDS_LOAD_ENV` setting determines the default save directory as follows:

Setting	Default Save To File
False	<code>~/ .cdsenv</code>
CWD	<code>currentWorkingDirectory/.cdsenv</code>
addCWD	<code>currentWorkingDirectory/.cdsenv</code>
CWDElseHome	<code>currentWorkingDirectory/.cdsenv</code> if it exists, else <code>HOME/.cdsenv</code>

- ❑ 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

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 fails to run.

To specify colors used in the CIW, add one of the following `ciw` environment variables to your `.cdsenv` file for each color you want to specify:

`ciwWarnColor`

`ciwErrorColor`

`ciwMatchParenColor`

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

`ciwMismatchParenColor`

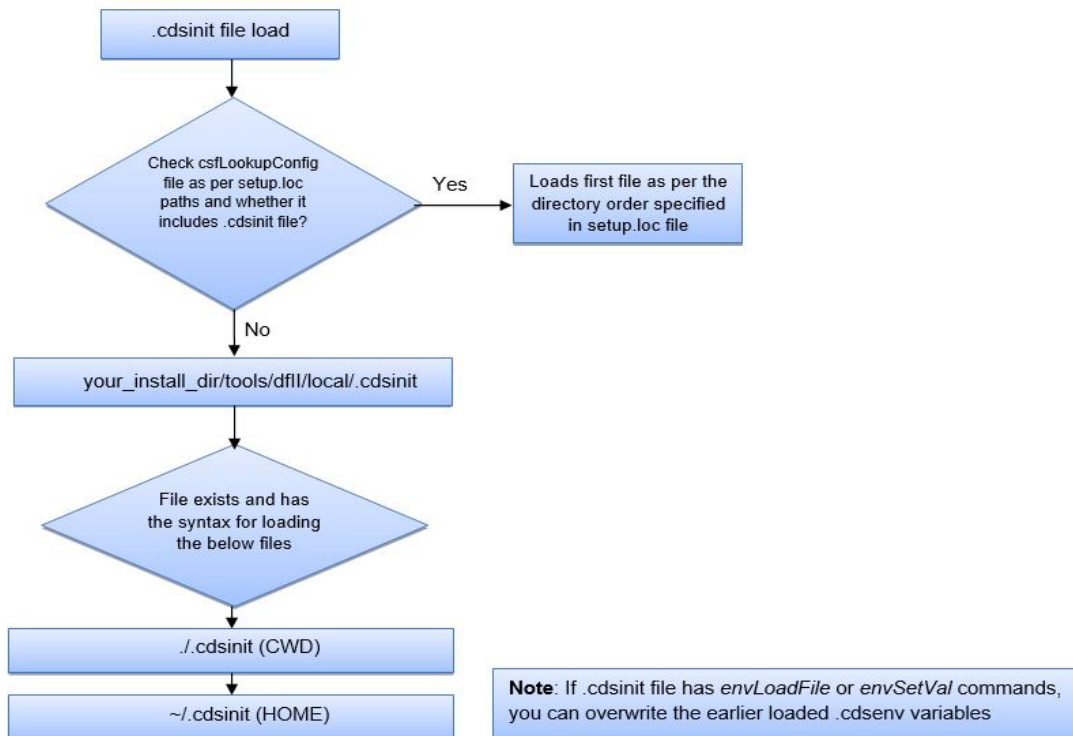
`ciwMatchCmdColor`

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).

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`.

3. `.cdsinit` (initialization instructions)

Flow chart `.cdsinit` file



Executable SKILL commands in your `.cdsinit` file takes precedence over settings in your `.cdsenv` file if you have both files in your home or workarea directory.

4. `display.drf` (display resource file)

The display resource files get loaded in the specified order:

- ☐ The Cadence-supplied default display resource file

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

`your_install_dir/share/cdssetup/dfII/default.drf`

This file is used with the Virtuoso Schematic Composer.

- ❑ A local display resource file you specify using the `drfPath` variable in your `.cdsenv` file.

`graphic drfPath string "path/display.drf"`

Use this optional file to provide required display resource definitions. Naming the file as `display.drf` is recommended but not required.

- ❑ Optional site `$CDS_SITE/display.drf` and project `$CDS_PROJECT/display.drf` display resource options files. Your system administrator can place these files in the site and project directories, if those directories are set active at your site. These files must be called `display.drf`.

- ❑ Personal display resource file (`~/display.drf`)

This is an optional file that you can customize and place in your home directory. This file must be named as `display.drf`.

- ❑ The current directory (`./display.drf`)

This is an optional file that you can customize and place in the directory from which you are initiating the software. This file must be named as `display.drf`.

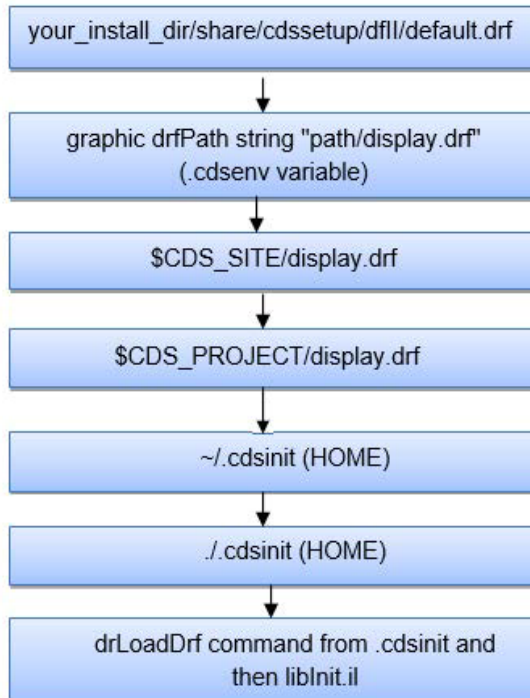
After initialization you can also load a `display.drf` file using SKILL function

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

```
drLoadDrf("/path/display.drf")
```

Flow chart display.drf file



Related Topics

[User Preference Options](#)

[Opening Cdsenv Editor in CIW](#)

[License Checkout Behavior in .cdsenv](#)

User Preference Options

You can set various options in your `.cdsinit` file or to the CIW input line using the commands in the following table:

Note: Before setting these `ciw->` commands, you must add `ciw = hiGetCIWindow()` in `.cdsinit` to return the window identity of the CIW.

To change	Command
Scroll bars	<code>ciw->useScrollbars = {t nil}</code>
Infix	<code>ciw->infix = {t nil}</code>
Option forms displayed when commands start	<code>ciw->expertMode = {t nil}</code>
Undo limit	<code>hiSetUndoLimit(<i>x_limit</i>)</code> Valid values: Integer between 0 and 128, where 0 disables undo
Nest limit	<code>ciw->nestLimit = <i>x_nestLimit</i></code> Valid values: Integer between 1 and 20 The default is 5.
Double-click time	<code>hiSetMultiClickTime(<i>x_msec</i>)</code> Valid values: Integer number of milliseconds
Display mouse bindings	<code>ciw->displayMouseBinding = {t nil}</code>
Text font	<code>hiSetFont("text" "<i>t_fontName</i>")</code> You can use * as a wildcard in any position. For example: <code>hiSetFont("text" "-*-courier-medium-r-*-*-12-*")</code>
Focus to cursor	<code>ciw->focusToCursor = {t nil}</code>
Form placement	<code>ciw->formPlacement =</code> <code>{ "top" "bottom" "left" "right" "center" "default" }</code>
Form placement relative to	<code>ciw->formRelativeTo = { "CIW" "screen" "currentWindow" }</code>
Option form placement	<code>ciw->optionFormPlacement =</code> <code>{ "top" "bottom" "left" "right" "center" "default" }</code>
Option form placement relative to	<code>ciw->optionFormRelativeTo =</code> <code>{ "CIW" "screen" "currentWindow" }</code>

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

To change	Command
Warp pointer to dbx	<code>ciw->warpPointer = {t nil}</code>
Show menu bindkeys	<code>ciw->displayMenuBindkeys = {t nil}</code>

Related Topics

[Specifying User Preferences](#)

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, 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)
```

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

Related Topics

[hiSetFilterOptions](#)

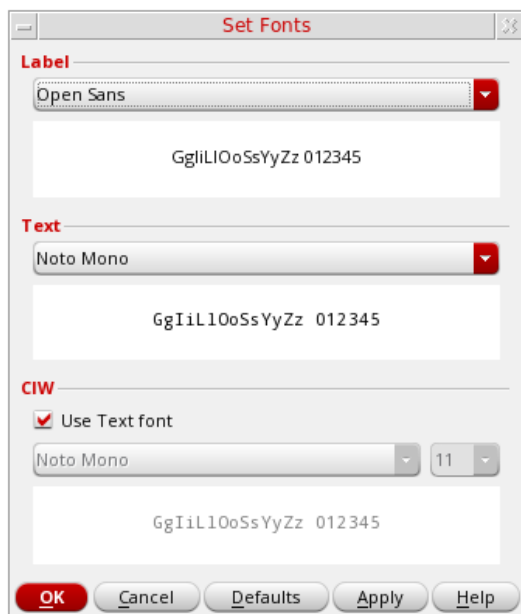
Viewing the Font List and Setting Fonts

To view the list of fonts available on your system:

† 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.



XFonts have been replaced by TrueType fonts that improve the general appearance and readability of the user interface.

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.

Related Topics

[hiSetFont](#)

GUI Default Setup Using `.cdsinit`

The `.cdsinit` file enables you to define the default settings for a form. The form can then be preloaded with the default settings even on first launch.

The `.cdsinit` file is written in the SKILL language. It uses environment variables to define the default settings for various form fields. When the `.cdsinit` file is called to load the form defaults, the file retrieves the predefined values for the environment variables and populates the form accordingly.

The environment variables defined in the `.cdsinit` file are accessed using the `getShellEnvVar` command. If the command returns a value `nil`, it indicates the environment variable is not defined. Therefore, no default settings are available for the associated form field.

Related Topics

[User Preference Options](#)

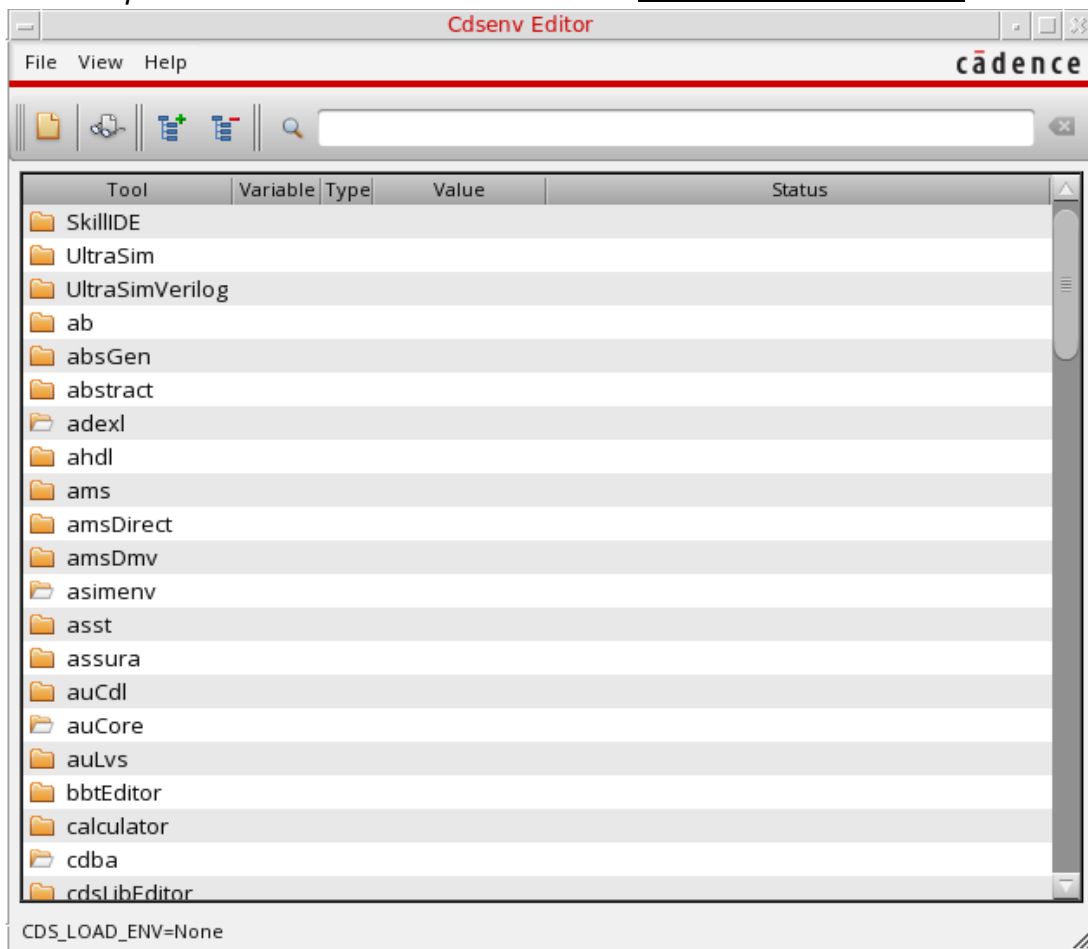
[Log Filter Options](#)

Opening Cdsenv Editor in CIW

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.

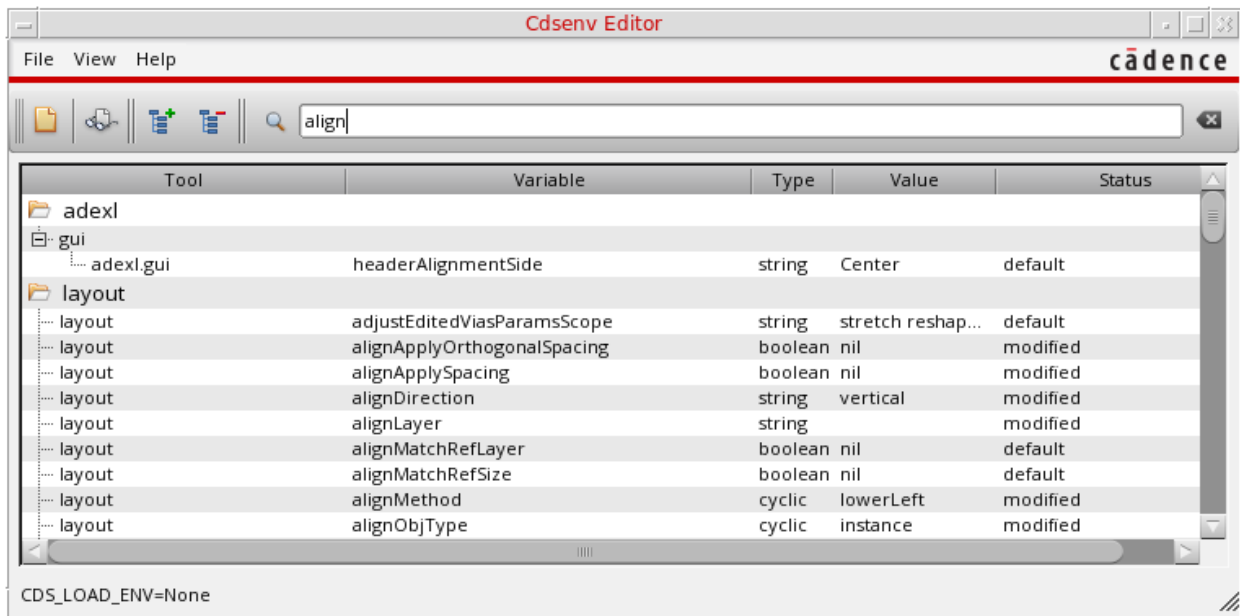
Related Topics

[Cdsenv Editor Form](#)

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 gets 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.

To view a list of all the environment variables, which are modified from their default value, type `modified from default` in the *Search* field.

Additionally, to copy the environment variables to clipboard, right-click on the variable you want to copy and select *Copy*, you can paste these values directly on CIW. You can copy multiple variables at a time.

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

Related Topics

[Save Defaults Form](#)

[Environment Settings in Virtuoso](#)

[floatPrecision](#)

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.

You can specify the following environment variables in the `.cdsenv` file to specify the order of checkout of available licenses.

- `VLSSLicenseCheckoutOrder`
- `VSELicenseCheckoutOrder`
- `VLSAdvOptLicenseCheckoutOrder`
- `EADLicenseCheckoutOrder`
- `maestroCheckoutOrder`
- `VIVALicenseCheckoutOrder`

For information on specifying these environment variables, see [CheckoutOrder Variables](#).

Use Next License Variables

The following `.cdsenv` variables can also be set to specify how to use the next available product license:

- `VSEL_UseNextLicense`
- `VLSSL_UseNextLicense`
- `VLSXL_UseNextLicense`

These environment variables can have the following values:

Value	Use
<code>prompt</code>	Displays the Next License dialog each time a requested license is not available. During a Virtuoso session, the Next License dialog gets displayed only once for each application.
<code>always</code>	Commands Virtuoso to always check out another license if the requested license is not available.
<code>never</code>	Commands Virtuoso to never check out another license.

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

The default setting for all of these variables is `prompt`.

For more information, see [UseNextLicense](#) and [Use Next License Dialog Box](#).

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
<i>Session</i>	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.
<i>Skip</i>	Skips the next license and tries to check out next license in check out order.
<i>Always</i>	Virtuoso always attempts to check out another license and the appropriate next license environment variable (for example, <code>VLSSL_UseNextLicense</code>) is updated in the <code>.cdsenv</code> (to <code>always</code>). The Next License dialog box is not displayed again when Virtuoso is restarted.
<i>Never</i>	Virtuoso does not try to check out another license and the appropriate next license environment variable (for example, <code>VLSSL_UseNextLicense</code> is updated in the <code>.cdsenv</code> to <code>never</code>). The Next License dialog box is not displayed again when Virtuoso is restarted.

Related Topics

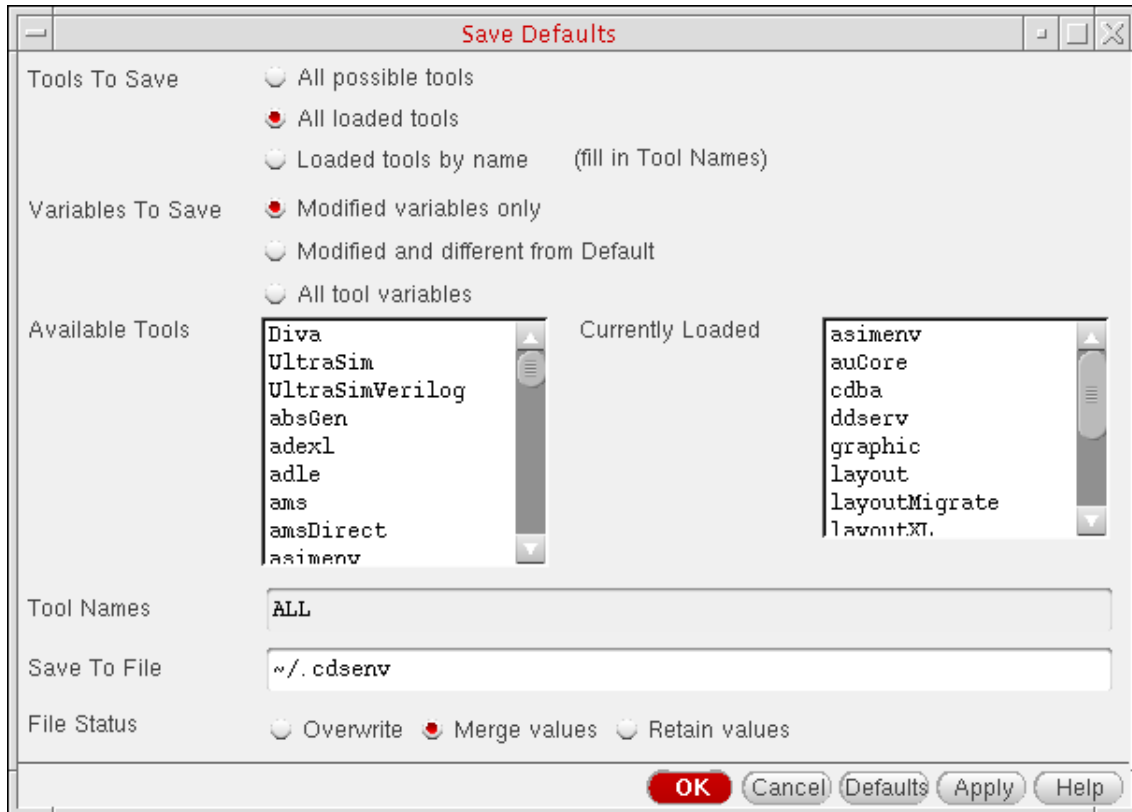
[Starting Virtuoso Studio](#)

Specifying New Default Values for a Virtuoso Session

To specify and save new default values from one session to the next:

1. From the Command Interpreter Window, choose *Options – Save Defaults*.

The Save Defaults form appears.



2. Select an appropriate option from the *Tools To Save* radio buttons to specify the save method for the environments.
3. Select an appropriate option from the *Variables To Save* radio buttons.

Radio buttons are not available if you selected *All possible tools* above.

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 an appropriate option from the *File Status* radio buttons.
6. Click **OK**.

Virtuoso Studio Design Environment User Guide

Environment Settings in Virtuoso

Settings are saved to the `.cdsenv` file and read the next time you start the software.

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.

Related Topics

[Editing an Environment Variable in Cdsenv Editor](#)

[Cdsenv Editor Form](#)

Virtuoso Studio Design Environment Launch

To launch and operate Virtuoso Studio Design Environment, you need to start the Cadence software and then you can use the command-line options to perform various operations. After setting up the above, you can launch Virtuoso.

Before you start Virtuoso:

- Obtain the required licenses and software.
- 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`).

Note: Both `.cdsenv` and `.cdsinit` are read by Virtuoso in a designated order.

Related Topics

[Starting Cadence Software](#)

[Command-Line Options For Cadence Applications](#)

[Backing Up Your Work](#)

[Unlocking an Application](#)

[Get Help with Virtuoso Menus](#)

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:

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

```
cd <your_working_Directory>
```

In Cadence software, this directory becomes your current directory.

2. Set environment variables as necessary.
3. Type the command to start the Cadence software.
You can ask your system administrator for the command.

When Cadence software starts, it reads the `.cdsinit` file to set up your Cadence environment. The software then 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 `.cdsinit` file controls the reading functionality of the software.

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

Related Topics

[Opening a UNIX Window from the Library Manager](#)

[Command-Line Options For Cadence Applications](#)

Command-Line Options For Cadence Applications

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

Option	Description
-45	<p>Provides enhanced drawing of 45-degree diagonal lines. With this option, diagonal lines are not jagged.</p> <p>This behavior might vary from shell to shell.</p>
-help	<p>Lists the supported command-line options.</p>
-log fileName	<p>Specifies the log file (<i>fileName</i>) to record this session.</p> <p>For example, to save the log file of the <code>virtuoso</code> executable in the <code>myLogDir</code> directory using a name that reflects the date, such as <code>CDS23Jul16:35.log</code>:</p> <pre>virtuoso -log ~/myLogDir/CDS"date '+%d%h%H%M' ".log</pre>
-logtime	<p>Adds an absolute timestamp for every entry in the log file. 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.</p> <p>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 <code>CDS_USE_LOCAL_TIMESTAMP</code> to true.</p> <p>This environment variable also controls whether other timestamps in the <code>CDS.log</code> 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 <code>CDS_LOG_TIMESTAMPS</code> to true.</p> <p>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.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

Option	Description
<code>-logtimerel</code>	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.
<code>-maxload</code>	<p>Specifies the maximum number of CPU-bound tasks that can be executed simultaneously. The default value for <code>maxload</code> is the number of CPU cores in a machine.</p> <p>The <code>maxload</code> value should not exceed the maximum number of threads allowed in a Virtuoso session, which is either defined by <code>setrlimit</code> or by the <code>-maxthreads</code> option.</p> <p>A task is not 100% CPU-bound if it takes less than 1 CPU core load.</p>
<code>-maxthreads</code>	<p>Specifies the maximum number of threads that can be created in a Virtuoso session. The default value for <code>maxthreads</code> is driven by the hard limit set by <code>setrlimit</code>.</p> <p>If the number of threads in a session exceeds the <code>maxthreads</code> value set by the hard limit, the application exits and a warning message is displayed.</p>
<code>-noblink</code>	<p>Disables blinking. When blinking is disabled, objects that would normally blink in graphics windows stops blinking.</p> <p>In many cases, disabling blinking can improve graphics performance, particularly for many VNC X Window setups.</p> <p>You can also use the SKILL function <code>hiEnableBlink</code> to enable or disable blinking while you are running the application and <code>hiIsBlinkEnabled</code> to check if blinking is enabled.</p>
<code>-nocdsinit</code>	Does not read any <code>.cdsinit</code> files.
<code>-nograph</code>	<p>Specifies non-graphics mode that you can use for SKILL programming and replay log files. To use this mode, you must install the <code>Xvfb</code> server at <code>/usr/bin</code>. This option is used internally by Virtuoso ADE Explorer and Virtuoso ADE Verifier.</p> <p>This option is intended only for replaying log files generated in graphics mode for testing purposes.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

Option	Description
-nographE	<p>Emulates the non-graphical mode using an X server other than <code>cdsXndx</code> (the Cadence-supplied non-graphical X server). This option is similar to <code>-nograph</code> except that it uses the default display or the display specified with either the <code>DISPLAY</code> environment variable or the <code>-display</code> command line option. Windows and forms get drawn on the specified display.</p> <p>You can also start the non-graphical emulation mode by setting the following environment variable:</p> <p><code>CDS_NOGRAPH_DISPLAY display</code></p> <p>You can use either the <code>-nograph</code> or <code>-nographE</code> option to start the application.</p> <p>If you use both <code>-display</code> and <code>CDS_NOGRAPH_DISPLAY</code>, then the <code>-display</code> value overrides that of <code>CDS_NOGRAPH_DISPLAY</code>.</p>
-nographN	<p>Starts the non-graphical mode using the display specified by the argument <code>N</code>, where <code>N</code> can be a number 0 through 9 or <code>+</code>. The <code>+</code> option displays the application in the <code>:400</code> through <code>:409</code> display range.</p> <p>Do not use the <code>-nograph+</code> option with any of the <code>tenbase</code> options. Use the <code>-nographN</code> option instead.</p>
-nologtime	<p>Overrides the setting made by the environment variable <code>CDS_LOG_TIMESTAMPS</code> and disables timestamp logging.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

Option	Description
<code>-noxshm</code>	<p>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.</p> <p>Use the <code>-noxshm</code> option in the following situations:</p> <ul style="list-style-type: none">■ 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 <code>BadAccess</code> and <code>BadShmSeg</code> error messages when using Virtuoso graphically <p>Virtuoso automatically stops using the <code>-noxshm</code> option if shared memory access fails to work.</p>
<code>-replay fileName</code>	<p>Executes a SKILL file (<i>fileName</i>).</p> <p>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.</p>
<code>-restore fileName</code>	<p>Restores a saved session from <i>fileName</i>.</p> <p>This option cannot read a log file because it uses the SKILL function <code>load</code> to load the file toward the end of the initialization process.</p>
<code>-showCtrlFile</code>	<p>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.</p> <p>To view the control file content, you need to run the following command:</p> <pre>virtuoso -showCtrlFile</pre> <p>After you use this option, the Virtuoso session is terminated.</p>
<code>-V</code>	<p>Returns the release number, such as 4.4, without starting the Cadence software.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

Option	Description
<code>-W</code>	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. <code>your_install_dir/tools/dfII/bin/virtuoso -W</code>

Note: The software uses a 24-bit TrueColor visual. If it is not available, then 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 stops the operation.

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

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

Related Topics

[hiEnableBlink](#)

[hilsBlinkEnabled](#)

[Starting Cadence Software](#)

Starting Virtuoso Studio

To start Virtuoso Studio from a UNIX prompt:

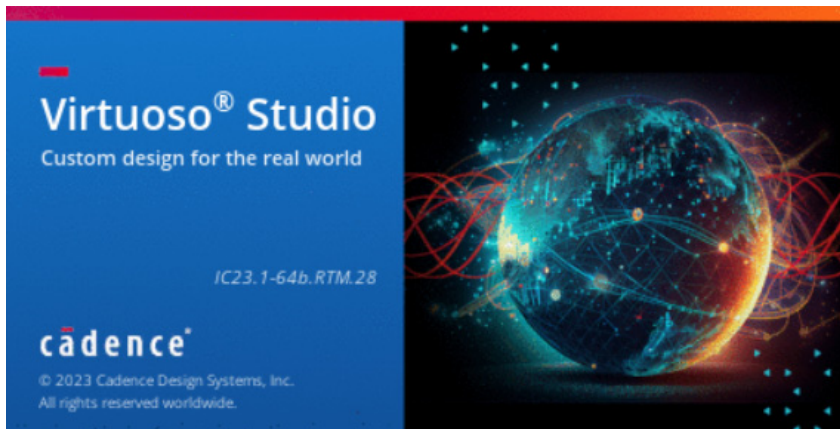
1. Change to the working directory from where you want to start the software:
`cd <your_working_directory>`
2. Confirm if the Virtuoso Studio executable is set properly: `which virtuoso`
3. Define the design library and reference libraries (`cds.lib`).
4. Type the command to start the Virtuoso Studio software: `Virtuoso &`

Examples of starting a Cadence workbench follow:

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

```
virtuoso &
```

The splash screen is displayed before the CIW opens.

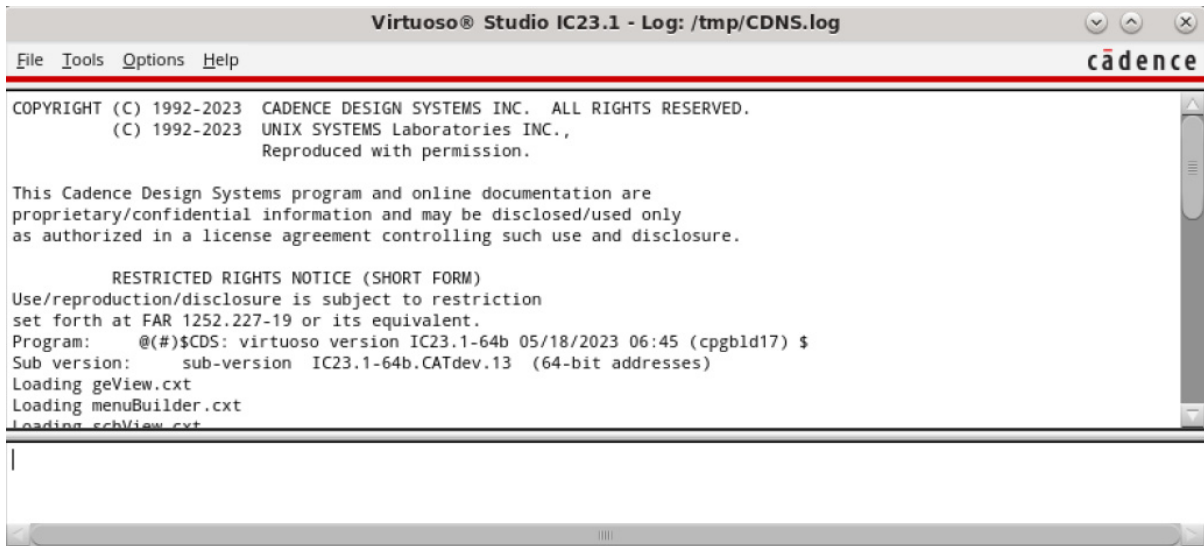


To disable the displaying of the splash screen use the shell environment `CDS_NOSPLASH` the command line option `-nosplash`.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

The CIW appears.



- ➔ To start Virtuoso Studio as a background process, with outputs being sent to a log file, type the following command in CIW:

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

In this case, output sent to the terminal also gets written to the log file, which can be accessed later, even if Virtuoso Studio 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 Studio using the **screen** command as follows:

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

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

- ➔ To start Cadence software in 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
```

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

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

Related Topics

[Environment Settings in Virtuoso](#)

[Viewing the cds.lib Updates](#)

Saving Changes in Virtuoso Studio Design Environment

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

To save design work:

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

To save changes to session defaults:

- ➔ From the CIW, choose *Options – Save Defaults*.

The Save Defaults form appears.

To save the current position of windows and forms:

- ➔ From the CIW, choose *Options – Save Session*.

The Save Session form appears.

Related Topics

[Specifying New Default Values for a Virtuoso Session](#)

[Saving and Restoring Window Positions in Virtuoso](#)

[Save Defaults Form](#)

[Save Session Form](#)

[Saving and Renaming Log Files in Virtuoso Studio Design Environment](#)

Saving and Renaming Log Files in Virtuoso Studio Design Environment

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 at the start of 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.

Related Topics

[Saving Changes in Virtuoso Studio Design Environment](#)

[Save Cellviews Form](#)

Backing Up Your Work

To back up your work:

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 -)
```

It is recommended to back up your work regularly.

Related Topics

[Saving and Renaming Log Files in Virtuoso Studio Design Environment](#)

[Exiting Virtuoso](#)

Unlocking an Application

The use of locks in a multiuser environment is an important feature as we do not want multiple edits taking place at the same time. The `.cdslock` file is therefore vitally important for ensuring this.

However, in certain circumstances, such as, if Virtuoso exists unexpectedly causing some design files to stay locked, you may be required to remove an application lock.

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

1. At the command line, type,

```
% clsAdminTool
```

2. At the prompt, type either:

```
> -are . : To remove all locks from the current directory.
```

or

```
> -are <directoryLocation> : To remove all locks from the specified location.
```

Type `help` at the prompt to get a full list of the `clsAdminTool` options.

A list of the locks that have been removed gets displayed, for example:

```
BEGIN: Release Edit Locks.
```

```
/user/jsmith/CDS.log lrd880a 13660 1173285728
```

```
/user/jsmith/aspec/run/techManager.log kudos 10225 991131648
```

Related Topics

[Command-Line Options For Cadence Applications](#)

Exiting Virtuoso

You can exit Virtuoso by typing a command or selecting a menu item. The Design Environment prompt is available in both graphics and non-graphics modes. In graphics mode, a command prompt is available on the input line of the CIW.

Before you quit Virtuoso, you should save any design or data changes you made during this session. For more information, see [Saving Changes in Virtuoso Studio Design Environment](#).

To exit Virtuoso from the Design Environment prompt:

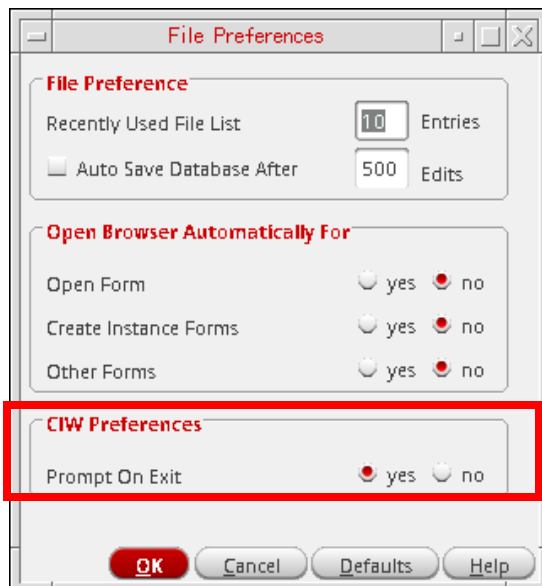
- ➔ Either on the CIW input line or at the nongraphics-mode Design Environment prompt, type `exit`.

To exit Virtuoso from a menu:

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

An Exit prompt appears.

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.



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

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

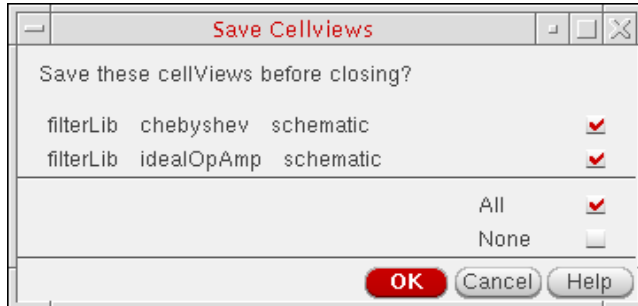
2. Click **Yes**.

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

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

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



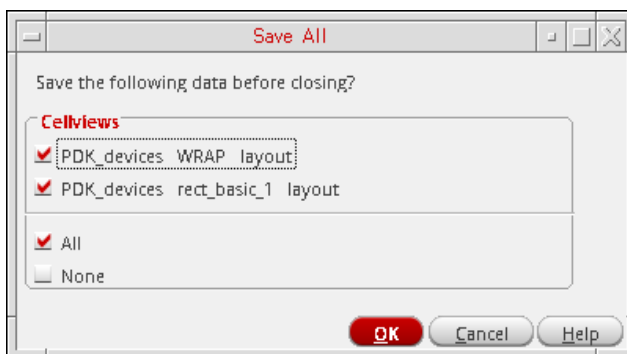
- Deselect the check boxes corresponding to the cellviews for which the changes do not need to be saved or click *None* to deselect all cellviews and save none of your changes. This is optional.
- Click *OK*.

The specified changes are saved and the application exits.

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, and techfiles, which are active in your session then you get prompted by the Save All form to choose *All* or *None*, without the list of modified data being displayed as it exceeds displayable limits.



Related Topics

[Save Cellviews Form](#)

[File Preferences Form](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

Virtuoso SKILL Interface

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Launch

Command Interpreter Window

After you launch Virtuoso software, you are in the Virtuoso Studio Design Environment. You can interact with the Virtuoso Studio Design Environment using 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.

Related Topics

[File Menu in the CIW](#)

[Tools Menu in the CIW](#)

[Options Menu in the CIW](#)

[Get Help with Virtuoso Menus](#)

[Session Commands and System Responses](#)

[Mouse Controls in the Virtuoso Studio Design Environment](#)

File Menu in the CIW

Use this menu to perform various operations on CIW.

Menu	Description
<i>New</i>	Displays a submenu of the following items: <ul style="list-style-type: none">■ <i>Library</i> displays the New Library form. Alternative New Library forms gets displayed dependent upon whether the form was accessed from the CIW or directly from the <i>Cadence Library Manager</i>.■ <i>Cellview</i> displays the New File Form.
<i>Open</i>	Opens a cellview from the CIW.
<i>Import</i>	Displays a submenu of available translators for importing on your system.
<i>Export</i>	Displays a submenu of available translators for exporting on your system.
<i>Refresh</i>	Refreshes the design data and the CDF data.
<i>Make Read Only</i>	Lets you select one or more cellviews to make read-only.
<i>Bookmarks</i>	Displays a list of recently viewed files (design views).
<i>Recently Viewed Design List</i>	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.

Related Topics

[Creating a New Cellview](#)

[Opening a Cellview from the CIW](#)

[Importing and Exporting Designs](#)

[Library Manager Toolbar](#)

[Managing Bookmarks](#)

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

Creating a New Library in the Library Manager

importDuplicateBookmarks

Tools Menu in the CIW

The *Tools* menu provides access to commands to start the Library Manager and your design software and utilities. The commands available on this menu varies with the binary you are running.

Menu	Description
<i>Library Manager</i>	Opens the Library Manager.
<i>Library Path Editor</i>	Opens the Library Path Editor.
<i>SystemVerilog</i>	Opens the SystemVerilog Netlister form.
<i>NC-Verilog</i>	Opens the Virtuoso Verilog Environment for NC-Verilog.
<i>VHDL Toolbox</i>	Opens the VHDL Toolbox.
<i>ADE Assembler</i>	Opens ADE Assembler by creating a new view or opening an existing view.
<i>ADE Explorer</i>	Opens ADE Explorer by creating a new view or opening an existing view.
<i>ADE Verifier</i>	Opens ADE Verifier by creating a new view or opening an existing view.
<i>VIVA XL</i>	Opens Virtuoso Visualization and Analysis XL window to plot and analyze the simulation results.
<i>AMS</i>	Displays a submenu of commands for setting options and netlisting for the Cadence AMS simulator.
<i>Technology File Manager</i>	Opens the Technology File Tool Box window that lets you compile, store, and edit technology data.
<i>Display Resource Manager</i>	Opens the Display Resources Tool Box.
<i>Abstract Generator</i>	Opens the Abstract Generator form.
<i>Print Hierarchy Tree</i>	Displays a list showing the hierarchy of cellview instances in the selected cellview.
<i>Set Cell Type</i>	Displays the Set Cell Type form.
<i>Check Pin Accessibility</i>	Opens the Pin Accessibility Checker form.
<i>Migrate</i>	Displays a submenu of commands that operate on various Schematic and Layout Migration operations.

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

Menu	Description
<i>CDF</i>	Displays a submenu of commands that operate on Component Description Format.
<i>Diagnostic Center</i>	Opens the Diagnostic Center form.
<i>SKILL IDE</i>	Opens the Cadence SKILL IDE window.
<i>SKILL API Finder</i>	Opens the SKILL Finder window.
<i>Conversion Tool Box</i>	Opens the Conversion Tool Box.
<i>Uniquify</i>	Displays the Uniquify form.
<i>Virtuoso Multi Technology</i>	Opens the Virtuoso Multi Technology Enablement form.

Related Topics

[Setting the Cell Type from the CIW](#)

[Uniquify Form](#)

[Integrated Abstract Generator](#)

[Layout Editor Basics](#)

[SKILL Integrated Development Environment](#)

[CIW CDF Commands](#)

Options Menu in the CIW

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

Menu	Description
<i>Save Session</i>	Opens the Save Session form.
<i>Save Defaults</i>	Opens the Save Defaults form.
<i>User Preferences</i>	Opens the User Preferences form
<i>File Preferences</i>	Opens the File Preferences form.
<i>Log Filter</i>	Opens the Set Log File Display Filter form.
<i>Bindkeys</i>	Opens the Bindkey Configuration form.
<i>Fonts</i>	Opens the Set Fonts form.
<i>Cdsenv Editor</i>	Opens the Cdsenv Editor form.
<i>License</i>	Opens the Software Product License Management form.
<i>Checkout Preferences</i>	Opens the Auto Checkout Preferences form.
<i>Checkin Preferences</i>	Opens the Auto Checkin Preferences form.

Related Topics

[Saving and Restoring Window Positions in Virtuoso](#)

[Specifying New Default Values for a Virtuoso Session](#)

[Specifying User Preferences](#)

[Changing the Log Filter Options](#)

[Configuring Application Bindkeys](#)

[Viewing the Font List](#)

[Checkout Preferences](#)

[Checkin Preferences](#)

[Opening Cdsenv Editor in CIW](#)

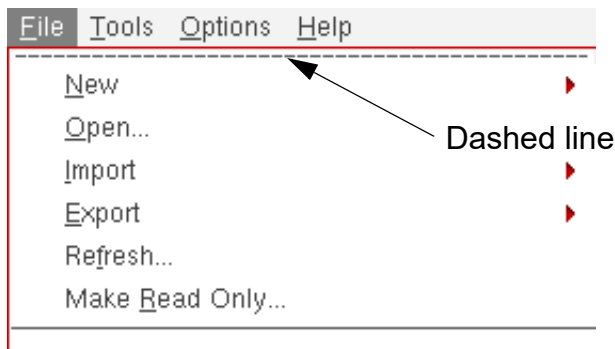
Tearing Off a Menu in the CIW

You can place menus on desktop for easy access by enabling the tear off menu feature in the CIW. You can also continue to access these menus from the menu banner in your application window.

To tear off a menu:

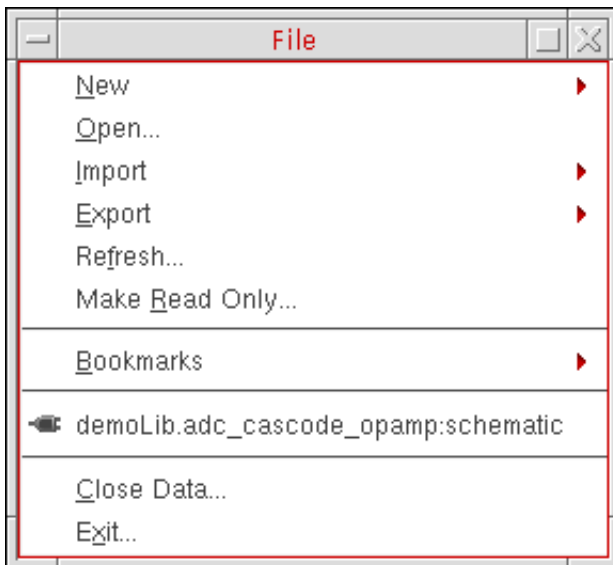
1. In the CIW window, click a menu item.

The corresponding Menu commands appear. 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 control aspects of a menu window using the control menu and other controls in the title bar.

Related Topics

[Opening Cdsenv Editor in CIW](#)

[Moving and Resizing a Menu Window](#)

Moving and Resizing a Menu Window

To move a menu window:

- ➡ 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.

To resize a menu window:

- ➡ 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).

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

The corresponding menu window gets resized with the specified size.

Alternatively, do the following:

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

The control menu appears.

2. From the control menu, select the size/resize option.

The menu frame appears along with a sizing cursor ()

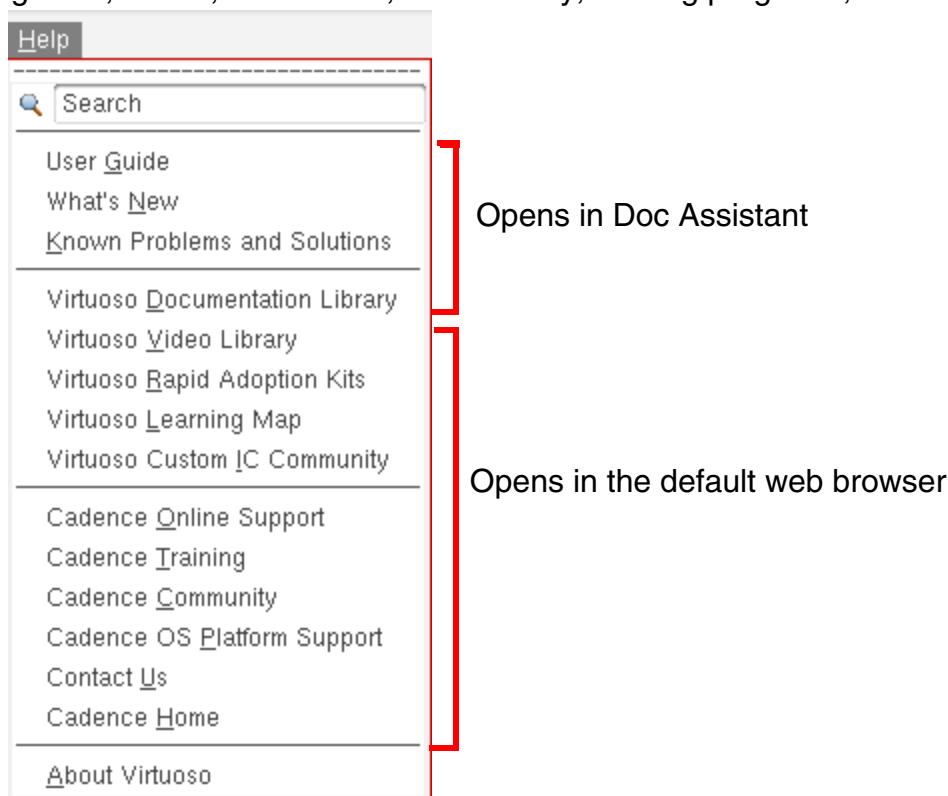
3. Drag-and-drop any corner of the menu window to set the size.

Related Topics

[Tearing Off a Menu in the CIW](#)

Get Help with Virtuoso Menus

There are various items you can access from the *Help* menu. You can access content like user guides, KPNS, what's new, video library, training programs, and so on.



Menu	Description
<i>Search</i>	A text field that lets you enter a search string. Press <code>Enter</code> to view the search results. Do not enclose the search string in double quotes.
<i>User Guide</i>	Opens the CIW documnetation available in the Virtuoso Studio Design Environment User Guide on Doc Assistant.
<i>What's New</i>	Opens the Virtuoso What's New document in Doc Assistant.
<i>Known Problems and Solutions</i>	Opens the Virtuoso Known Problems and Solutions document in Doc Assistant.

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

Menu	Description
<i>Virtuoso Documentation Library</i>	<p>Opens the Doc Assistant home page, which provides quick access to the following resources:</p> <ul style="list-style-type: none">■ Getting Started■ How-To Guides■ Reference■ Articles and App Notes■ Browse All Manuals■ Browse All Topics
<i>Virtuoso Video Library</i>	<p>Opens the Video Library page available on Cadence Learning and Support portal. This page lists the videos available for various Virtuoso products.</p> <p>You must have a registered account to access the content on the Cadence Learning and Support portal. Contact your IT support to ensure that the Internet ports required for video playback are enabled.</p>
<i>Virtuoso Rapid Adoption Kits</i>	<p>Opens the Rapid Adoption Kits page on the Cadence Learning and Support portal. 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.</p>
<i>Virtuoso Learning Map</i>	<p>Lists domain-specific training available on Cadence Training Services.</p> <p>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.</p>
<i>Virtuoso Custom IC Community</i>	<p>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.</p>

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

Menu	Description
<i>Cadence Online Support</i>	<p>Opens the Cadence Learning and Support portal, 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 the Cadence Learning and Support portal 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.</p> <p>You can also access the Cadence Learning and Support portal by clicking the Cadence logo available in the upper-right banner in each Virtuoso window.</p> <p>To disable this behavior, you need set the following shell variable:</p> <pre>Setenv CDS_LOGO_ACTION OFF</pre>
<i>Cadence Training</i>	<p>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.</p>
<i>Cadence Community</i>	<p>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.</p>
<i>Cadence OS Platform Support</i>	<p>Provides information about the current Cadence software releases and the supported platforms.</p>
<i>Contact Us</i>	<p>Opens the Cadence Customer Support Contacts web page, which provides customer support contact information for different regions.</p>
<i>Cadence Home</i>	<p>Opens the Cadence corporate web site.</p>
<i>About Virtuoso</i>	<p>Displays Virtuoso Studio Design Environment version information.</p>

Related Topics

File Menu in the CIW

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

Tools Menu in the CIW

Options Menu in the CIW

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.

You can control the maximum number of previous commands that appear in the output area, see [User Preferences Form](#).

You can control what you see in the output area by changing the log filter. 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.

Related Topics

[Changing the Log Filter Options](#)

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:

L: schSingleSelectPt()	M: hiCloseWindow(getCurrentWindow())	R: schHiMousePopU
------------------------	--------------------------------------	-------------------

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.

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 the deprecated `XKeyCodeToKeysym` function. Also, update the event list to include new `GenericEvent`.

Related Topics

[Bindkeys and Access Keys](#)

[User Preferences Form](#)

Actions Required 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.

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.

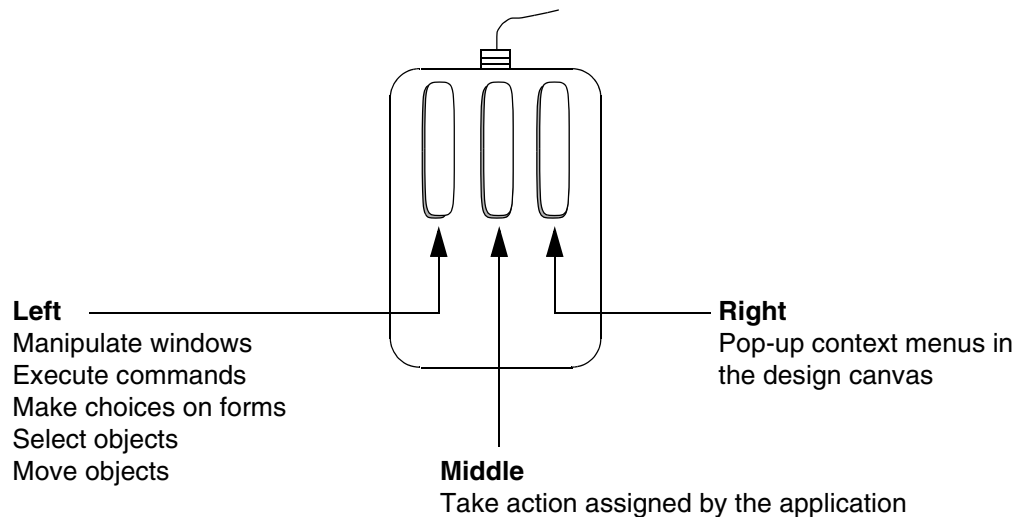
To cancel a command and remove the prompt, press the *Escape* key while the cursor is in the design window (the current, active window).

Related Topics

[Mouse Bindings Line](#)

Mouse Controls in the Virtuoso Studio Design Environment

You can use mouse buttons in the following ways:



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.

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	<code>hiSetBindKey(p "None<Btn4Down>" "geScroll(nil \"n\" nil) ")</code>
Scroll wheel down	Scroll/pan down	<code>hiSetBindKey(p "None<Btn5Down>" "geScroll(nil \"s\" nil) ")</code>

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

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

Related Topics

[Viewing the Current Bindkeys for an Application](#)

[Bindkey Configuration form](#)

[Bindkeys and Access Keys](#)

[geScroll](#)

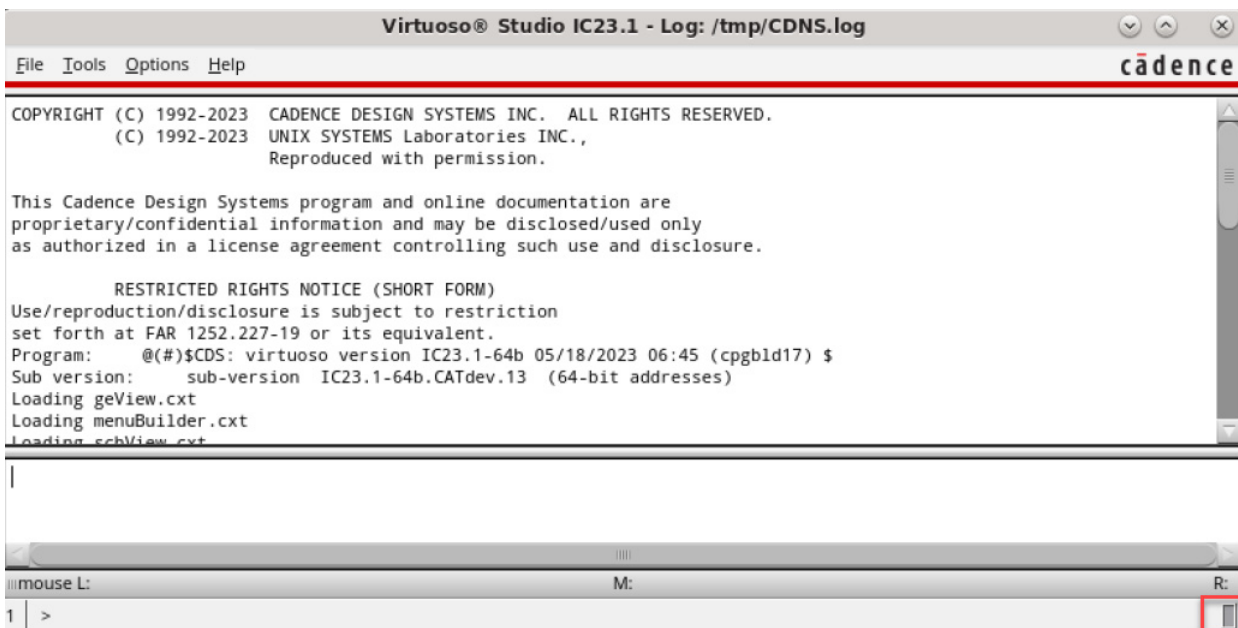
[hiSetBindKey](#)

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.

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.



Related Topics

[User Preferences Form](#)

[Resource Indicator Environment Variables](#)

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

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.

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:



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.

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

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 `swapActivity` environment variables.

You can use the `CDS_MEMPERF_WARNLOG` shell environment variable to run user-specified executables. For example, the following command makes 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 consists of several pairs of name and value. Any unrecognized names are ignored.
- `WHEN`
Specifies when the program should run.
- `WHEN=SwapActivityDialog`
Indicates the provided script runs when the *Swap Activity Warning* dialog box is displayed.
- Any unrecognized value of `WHEN` is silently ignored.
- `SCRIPT`
Specifies the program which is used and can be specified only once. The associated value consists 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.

Virtuoso Studio Design Environment User Guide

Command Interpreter Window

- 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 states that: *Another warning will be output after X minutes if the condition persists or recurs.*

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

Related Topics

[Design Environment Variables](#)

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](#).

Related Topics

- [Opening a Cellview from the CIW](#)
- [Closing Cellviews in Virtuoso](#)
- [Saving Modified Data](#)
- [Default Instance Prefixes](#)

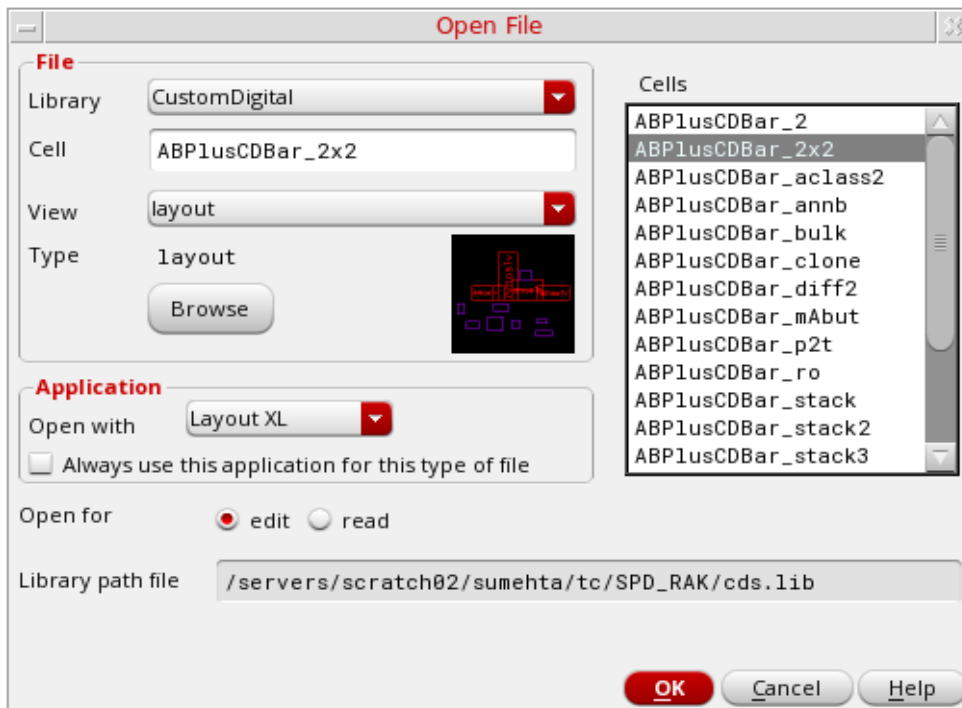
Opening a Cellview from the CIW

To open a cellview from the Command Interpreter Window or your session window:

1. Choose *File – Open*.

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

The Open File form appears



2. In the *File* group box:
 - 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.

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

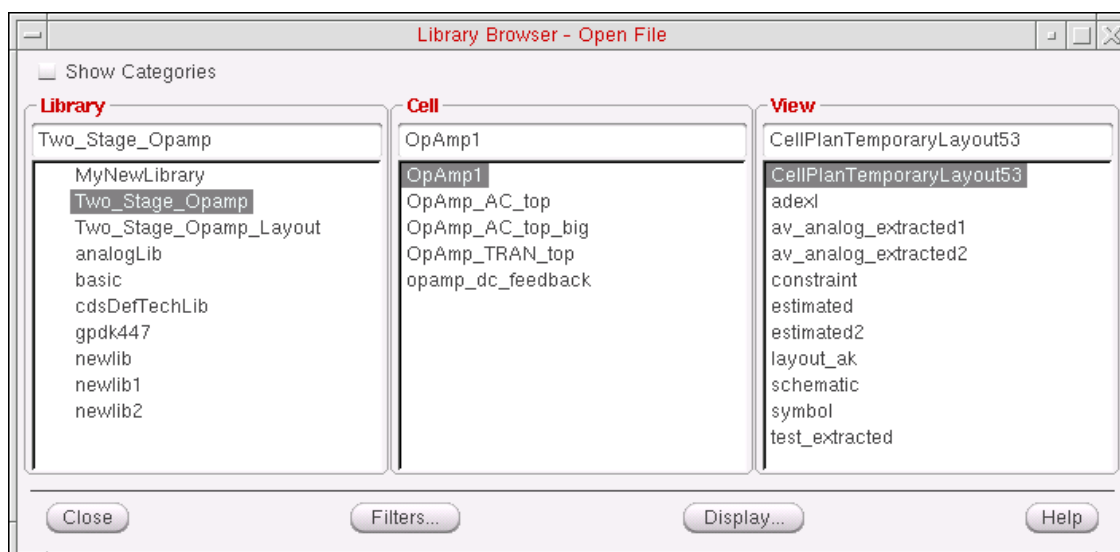
Once the cell name is entered in the *Cell* field, the views available for that cell appears in the *View* drop-down combo box.
 - c. In the *View* drop-down combo box, select a view name.

Virtuoso Studio Design Environment User Guide

Working with Cellviews

If a thumbnail of the selected cellview is available, this gets displayed in the Open File form as an aid to selection. If no thumbnail is available, the space remains blank. For more information on thumbnails, see [Thumbnail Images of Cellviews](#).

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



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

For schematic cellviews	<i>ADE Assembler</i> <i>ADE Explorer</i>
	<i>SMG Model Schematic</i> <i>Schematics L</i> <i>Schematics XL</i>
For symbols	<i>Symbol L</i> <i>Symbol XL</i>
For layout views	<i>Layout ViewerLayout XL</i> <i>Layout EXL</i> <i>Layout MXL</i>
For config views	<i>Hierarchy Editor</i>
For an ADE state view	<i>ADE State</i>
For text views	<i>Text Editor</i>

Virtuoso Studio Design Environment User Guide

Working with Cellviews

For more information, see [Starting Virtuoso Studio](#).

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

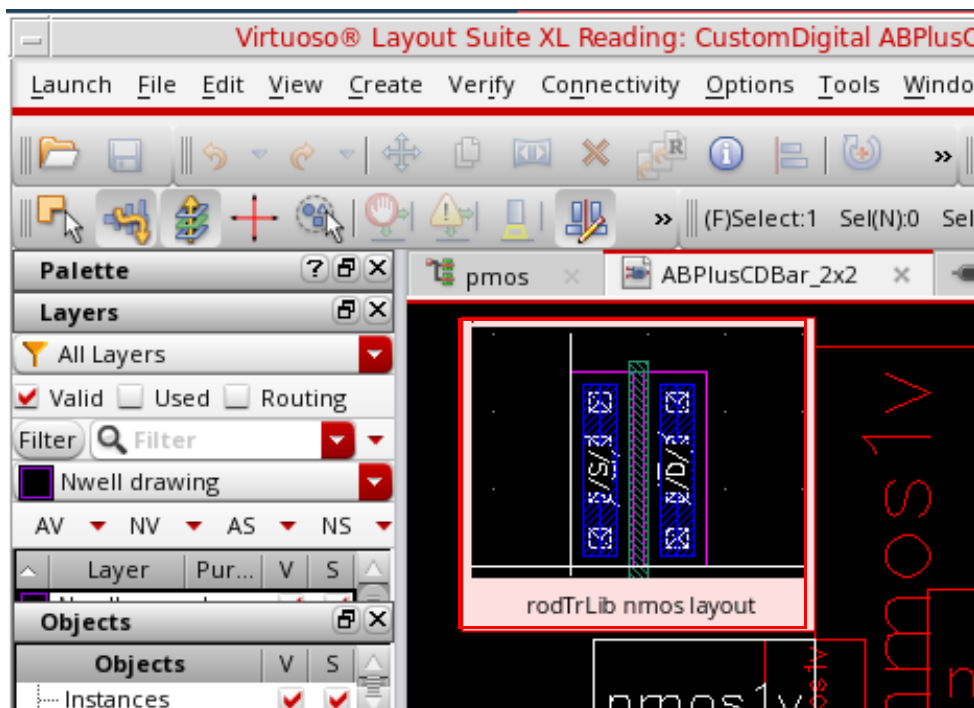
To open the file in edit mode, you must have write permission to the file.

6. 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.
7. 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.

Enable Design Preview

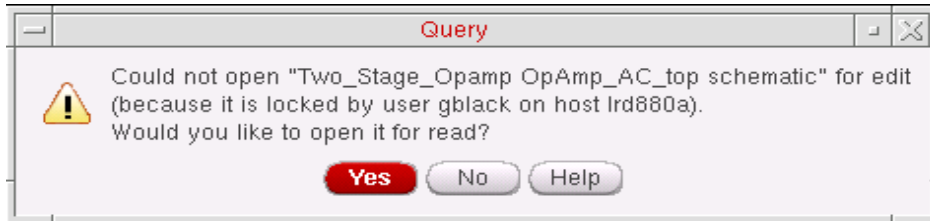
If multiple designs are open, place the mouse pointer on a tab to preview the design open in that tab, as shown below:



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 gets 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.



This warning message gets 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.

Related Topics

[Open File form](#)

[Log Filter Options](#)

[stopLevel](#)

[imageTabTip](#)

Cellview History

Virtuoso Studio 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 Studio 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.

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. Next time when you open this cellview, only Layout Viewer and Layout EXL gets displayed as the available choices to open the cellview and Layout XL is not available.

Related Topics

[Opening a Cellview from the CIW](#)

[Open File form](#)

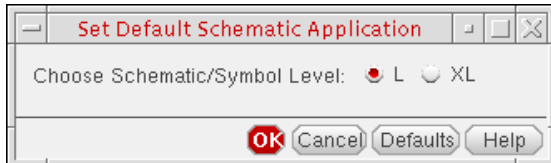
[ignoreAppTierInHistory](#)

Setting the Default Application for a Cellview Type

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

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*

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

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

- ☐ *XL*

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.

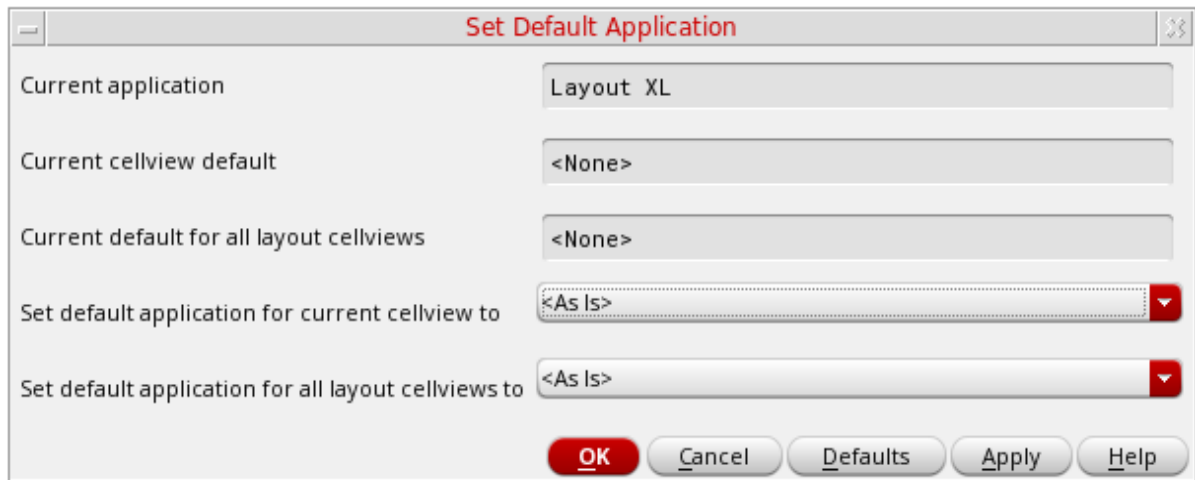
To set the default application for a layout cellview:

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

Virtuoso Studio Design Environment User Guide

Working with Cellviews

The Set Default Application form appears.



Set Default Application

Current application: Layout XL

Current cellview default: <None>

Current default for all layout cellviews: <None>

Set default application for current cellview to: <As Is>

Set default application for all layout cellviews to: <As Is>

Buttons: OK, Cancel, Defaults, Apply, Help

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.

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.

Related Topics

Copying the Current Cellview to a New Session Window

Closing Cellviews in Virtuoso

To close a cellview on a tab in a session window that has more than one tab:

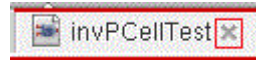
1. Right-click the tab you want to close.

A pop-up menu appears.

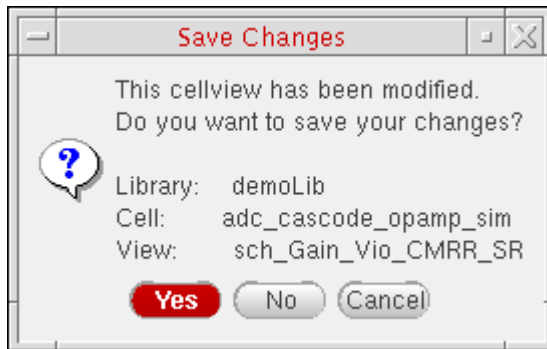
2. Choose *Close Tab*.

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.



- ☐ Click *Yes/No* to save or discard changes.

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

To close all cellviews except the one on the current tab in a session window that has more than one tab:

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.

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.

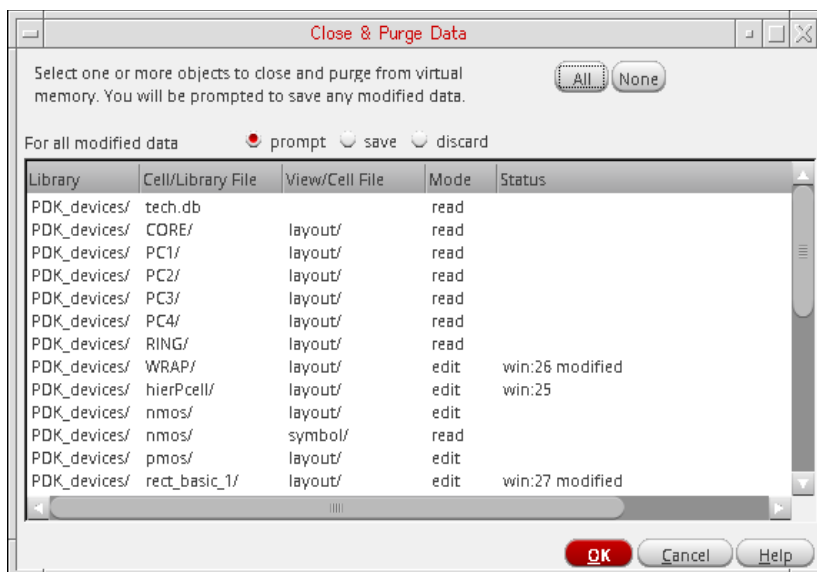
Instead of closing each cellview individually, you can close all cellviews at once as follows:

- ➔ In the CIW, choose *File – Close Data*.

Virtuoso Studio Design Environment User Guide

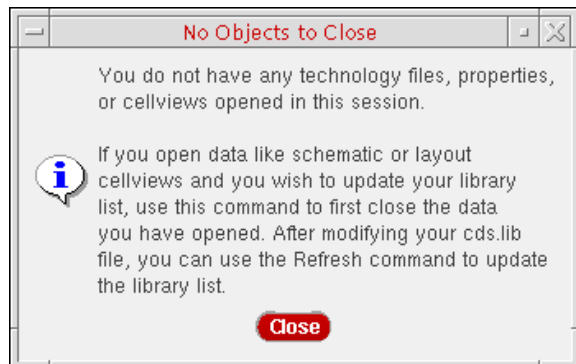
Working with Cellviews

The Close and Purge Data Form appears.



Note: You can sort the information in the Close and Purge Data form by clicking the column header of interest. When you click *Status*, first data that is open in a window and has been modified is listed. This is followed by data that has been modified, and then the rest of the data is listed.

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



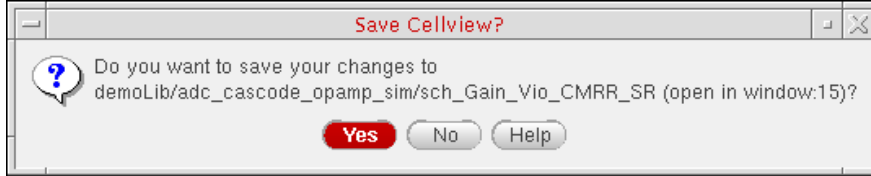
1. Choose how Virtuoso handles modified data on *File – Close Data* selection.
2. On the Close and Purge Data form, do one of the following:
 - ☐ Click *All/ None* on the form.
 - ☐ Click and drag to select a contiguous set of cellviews.
 - ☐ Control- click individual (noncontiguous) cellviews to select each one.

Virtuoso Studio Design Environment User Guide

Working with Cellviews

3. Click *OK*.

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



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

Related Topics

[Copying the Current Cellview to a New Session Window](#)

[Close and Purge Data Form](#)

[Session Windows and Workspaces](#)

Copying the Current Cellview to a New Session Window

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

- ➡ 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 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.

Related Topics

[Closing Cellviews in Virtuoso](#)

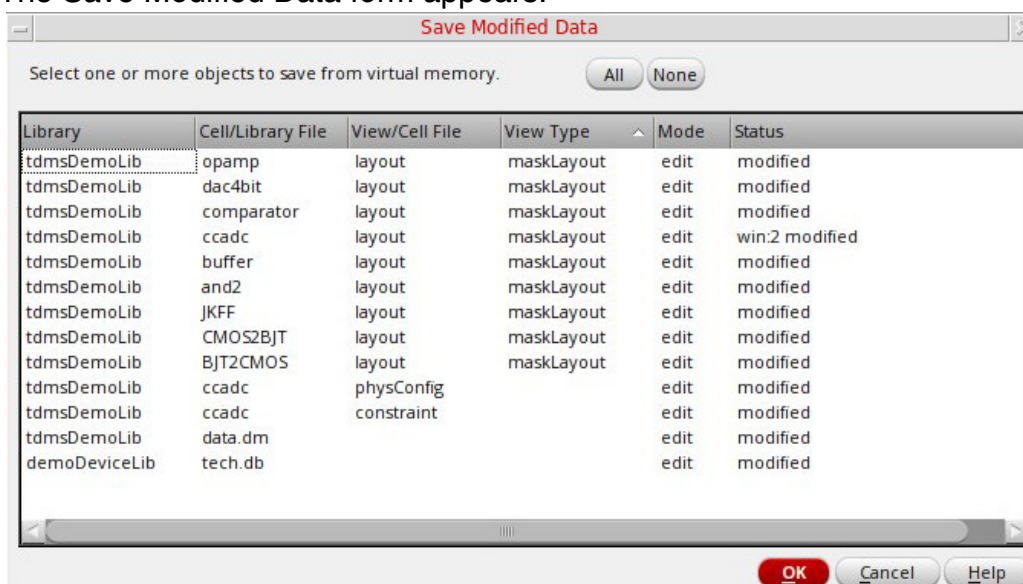
Saving Modified Data

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.



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 stays open for any further editing.

Virtuoso Studio Design Environment User Guide

Working with Cellviews

Related Topics

[Save Modified Data Form](#)

[ddsHiSaveData](#)

Default Instance Prefixes

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.

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:

<code>cdba</code>	<code>dbInstNamePrefix</code>	<code>string</code>	<code>"I"</code>
<code>cdba</code>	<code>dbArrayInstNamePrefix</code>	<code>string</code>	<code>"M"</code>

Related Topics

[`dbInstNamePrefix`](#)

[`dbArrayInstNamePrefix`](#)

[Changing the Prefix Using a Property](#)

Changing the Prefix Using a Property

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

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

The Edit Cellview Properties form appears.

The 'Edit Cellview Properties' dialog box is shown with the 'Property' tab selected. The fields are as follows:

Library	ether		
Cell	PLL_PFD		
View	layout		
k60%	((-0.4075 0) (42.4 31.48))		
Type	maskLayout	CBU per micron	2000
Mode	Edit	Save Needed	No
CellType	none		

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

The Add Property form appears.

The 'Add Property' dialog box is shown with the following fields:

Name	myProp		
Type	String		
Value			
Choices			
Minimum		Maximum	

String appears as the default *Type*.

Virtuoso Studio Design Environment User Guide

Working with Cellviews

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.

Related Topics

[dbArrayInstNamePrefix](#)

[dbInstNamePrefix](#)

Automatic Defragmentation of OpenAccess Databases

When objects are deleted from a cellview, the disk space is not immediately reclaimed. This is because it is more efficient and reliable to reuse the memory of the deleted objects than to deallocate and reallocate the memory. Undo needs to be able to restore objects to their original locations in memory and that cannot happen if the space has been reclaimed. Applications rely on knowing object locations in the memory and they will not function correctly if objects that are cached or are being accessed move in the memory.

Instead, space in the database is recovered (defragmented) when certain conditions are met and when the database is read from the disk. The database file size changes only when the database is saved after defragmentation has occurred. Defragmentation is an expensive operation, so it occurs only when a significant amount of space can be reclaimed. Defragmentation happens automatically and does not require any user intervention or application control.

Related Topics

[Opening a Cellview from the CIW](#)

[Saving Modified Data](#)

[Setting the Default Application for a Cellview Type](#)

Virtuoso Studio Design Environment User Guide

Working with Cellviews

Virtuoso Studio Design Environment Customization

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

You can customize specific application setup information by referring to the specific application documentation. Customization updates made via the CIW are common to multiple users or global to an entire site.

Additionally, 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. For more information, see [Environment Settings in Virtuoso](#).

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.

Area to be customized	Action files
UNIX paths, file locations, and device specifications	<code>.cshrc</code> , <code>.login</code> , or <code>.profile</code> file
General window appearance and behavior	<code>.Xdefaults</code> , <code>.xsession</code> or desktop/window manager configuration
Window appearance and command actions and environment variables for Cadence software	<code>.cdsinit</code> or <code>.cdsenv</code> file

Related Topics

[Customizing the Virtuoso Schematic Editor](#)

[Customizing the Layout XL Desktop](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

[Starting Virtuoso Studio](#)

[Bindkeys and Access Keys](#)

[Default Instance Prefixes](#)

[Specifying New Default Values for a Virtuoso Session](#)

.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 is always 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.

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

When CDMS (Cadence Directory Management System) is used by an application, it is expected that user data gets transferred. CDMS implicitly uses `.cadence` as the top-level directory name.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

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

File or Directory	Description
<code>history/userName.history</code>	Maintains information that appears in the History tree on the ADE Explorer Data assistant pane. You must not edit this file.
<code>Navigator</code>	Contains <code>Options.xml</code> where the program stores any options you specify on the Navigator Options form. You must not edit this file.
<code>toolbars</code>	Contains custom toolbar definition files.
<code>workspaces</code>	When you save a custom workspace, the program creates this directory along with an application-specific subdirectory containing your custom <code>.workspace</code> file; your <code>workspaces</code> directory might contain several application-specific subdirectories and each application-specific subdirectory might contain several <code>.workspace</code> files. <code>.workspace</code> files are binary files that you cannot edit.
<code>workspace.default</code>	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:

File or Directory	Content
<code>VLS-EXL</code>	<code>Modgen.patterns</code> pattern file for Modgen.
<code>cds.lib file</code>	Default library definitions.
<code>ci</code>	<code>config.xml</code> file defining constraint types for layout as well as a mapping between constraint type and parameter names and what the program displays for these.
<code>default.drf file</code>	Default display resource settings.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

File or Directory	Content
toolbars	byApplication subdirectory of Cadence-provided .toolbars files.
workspaces	Application subdirectories containing Cadence-provided .workspace definition files and workspace.default file.

Related Topics

[Working with Patterns](#)

[Cadence Library Definition](#)

[Display Resource Development and Usage](#)

[Virtuoso Workspaces](#)

[Using and Resetting Toolbars](#)

Specifying User Preferences

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

- ➔ From the CIW, choose *Options – User Preferences*.

The User Preferences form appears.

The screenshot shows the 'User Preferences' dialog box with the following sections and settings:

- Window Controls**
 - Mouse Prompts: Bottom
 - Side Dock Tabs: Bottom
 - Focus Policy: CanvasDelay
 - Focus Delay (ms): 200
 - Default Editor Background Color: [Color swatch]
 - Create New Window When Descending: ☐
 - Place Manually: ☐
 - Scroll Bars: ☐
 - Tear-Off Menus: ☒
 - Menu Shortcuts: ☒
- Command Controls**
 - Infix (No Click Necessary for First Point): ☐
 - Options Displayed When Commands Start: ☐
 - Web Browser: firefox
 - Undo Limit: 128
 - Nest Limit: 5
 - Double Click Time (ms): 200
 - Beep Volume: 0
- CIW Controls**
 - Input Area Lines: 1
 - Output Wrap Mode: Word or Anywhere
 - Retain Unique Commands: ☐
 - Output Area Lines: 1000
 - Syntax Highlighting: ☒
 - History In Place: ☒
 - Tab Stop: 4
 - Enter Key Executes Command: ☒
 - Raise CIW on: ☐
 - Warning: ☐
 - Error: ☐
- Dashboard Controls**
 - Display Dashboard Indicators: ☒
 - License Activity: CIW+Session

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

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 only takes place in the next Virtuoso session if you permanently save the changes in the `.cdsenv` file.

To make these changes permanent, choose *Options – Save Defaults*. For more information, see [Specifying New Default Values for a Virtuoso Session](#) or edit your `.cdsenv` file. Some of the changes get effective with the next command while others become effective the next time a window is opened.

Save your settings and apply them to future sessions:

1. In the Command Interpreter Window, 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.

Related Topics

[User Preferences Form](#)

[Save Defaults Form](#)

Saving and Restoring Window Positions in Virtuoso

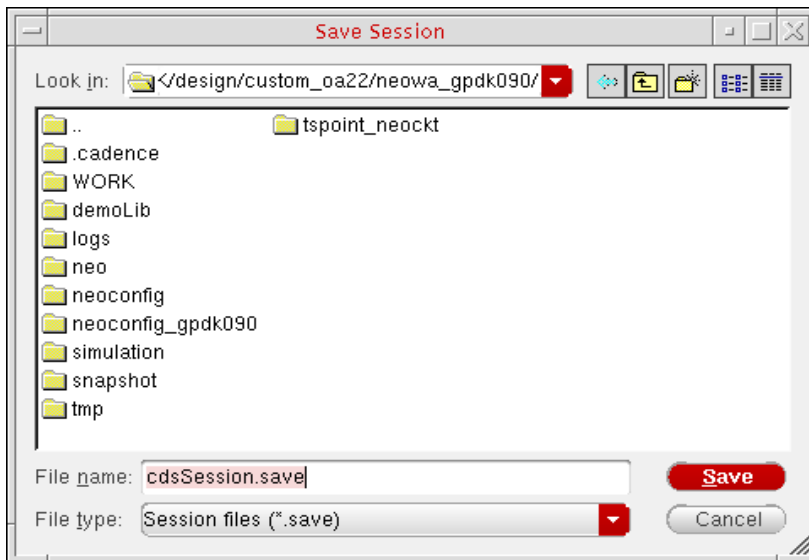
You can save the current position of design windows and forms. The exact information saved depends on your application.

Note: This does apply to workspace positioning.

To save window and form positions:

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. In the *Look in* field, change the directory location for the saved session file.
3. 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 of the CIW

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

- ➔ 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:

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

Saving and Restoring the Position of the CIW Using SKILL

To save and restore the position of the CIW:

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.

Related Topics

[hiResizeWindow](#)

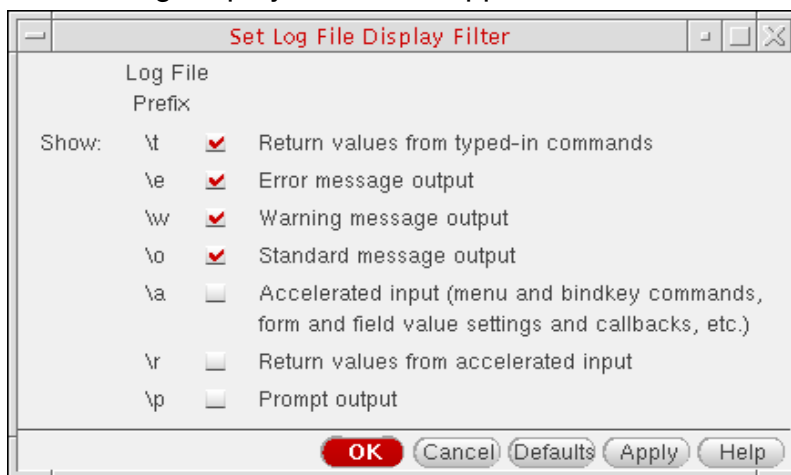
Changing the Log Filter Options

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:

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. Select check boxes 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.

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

Related Topics

Set Log File Display Filter Form

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Log Filter Options

Menu Banner Customization

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 Studio 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

Related Topics

[Customizing CIW Menus](#)

[Customizing the Virtuoso Schematic Editor](#)

[Customizing Menus](#)

Customize Toolbar Definition Files

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 *current folder_name/.cadence/dfII/toolbars/byApplication* so that the program finds and uses these definitions first. Use `setup.loc` to change the location of the files from current to other folder.

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
      )
    )
  )
  ...
)
```

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

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"
```


Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

```
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.

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.

Related Topics

[Cadence Setup Search](#)

[Using and Resetting Toolbars](#)

[Format for Main Toolbar Definition](#)

[Format for Item Type Action](#)

Format for Main Toolbar Definition

Use these mandatory and optional fields to define a toolbar:

Mandatory Field	Description
<code>name</code>	Specifies the name to be used in the creation of the toolbar. The name must be unique. Virtuoso Studio Design Environment automatically adds <code>_sessionWindowNumber</code> to the name.
<code>text</code>	Specifies the string to be used in the toolbar title and <i>ToolTip</i> .
<code>items</code>	Displays the list of item descriptions for the toolbar.

Optional Field	Description
<code>invisible</code>	Ensures that the toolbar is initially hidden on application launch.
<code>toolButtonStyle</code>	Specifies one of the following toolbar button styles: <code>textOnly</code> , <code>iconOnly</code> , <code>textBesideIcon</code> , or <code>textUnderIcon</code> . The default is <code>iconOnly</code> .
<code>getActionFunction</code>	Specifies a function that is called on all toolbar items of type <code>'getAction'</code> on this toolbar. The specified function must return an appropriate <code>hiAction</code> value. This functionality lets you reuse the existing <code>hiAction</code> values and share them amongst widgets. Therefore, if an <code>hiAction</code> is toggled using a widget, all other widgets that use the same underlying <code>hiAction</code> automatically reflects the correct state.

Related Topics

[Customize Toolbar Definition Files](#)

[Format for Item Type ComboBox](#)

[Format for Item Type Typein, Separator, and InheritToolbarsForm](#)

Format for Item Type Action

Use these mandatory and optional fields to specify various item type actions:

Mandatory Field	Description
<code>type</code>	A direct <code>action</code> (callback).
<code>name</code>	The name to be used in the creation of this <code>action</code> . The name must be unique among all <code>actions</code> .
<code>text</code>	Specifies text for <code>iconText</code> and tooltip.
<code>callback</code>	Specifies the actions when the <code>action</code> is processed.
<code>icon</code>	The name of the icon file (must be in <code>.png</code> format). This file is looked up in the standard <code>setup.loc</code> locations, prefixed by the path <code>icons/24x24</code> . The standard locations include <code><installdir>/share/cdssetup</code> , <code>./cadence</code> and <code>~/.cadence</code> . See Using and Resetting Toolbars .

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Mandatory Field	Description
<code>enableCondition</code>	<p>Sets the toolbar button enablement condition, where the following activation settings can be specified:</p> <ul style="list-style-type: none">■ <code>selected</code>: object is selected, button is enabled■ <code>modified</code>: cellview has been modified, button is enabled■ <code>editable</code>: cellview is editable, button is enabled■ <code>assistantsPresent</code>: assistants exist in cellview, button is enabled■ <code>notMultiSheet</code>: cellview is not part of a multi sheet, button is enabled■ <code>modifiedNotScratch</code>: identifies that the cellview is modified and not a scratch cellview <p>It can be a:</p> <ul style="list-style-type: none">■ 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 <p>The <code>enableCondition</code> can be set only for action types except for the <code>subAction</code> action item.</p>
<code>visibleCondition</code>	<p>Sets the visibility condition for a toolbar. Settings that can be specified are <code>comboBox</code>, <code>typein</code>, and <code>separator</code>.</p> <p>It can be:</p> <ul style="list-style-type: none">■ 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 <p>The <code>visibleCondition</code> can be set for any toolbar item type.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Optional Field	Description
<code>invisible</code>	Hides the <code>action</code> item initially.
<code>subAction</code>	Specifies the work for this <code>action</code> . You can choose from: <ul style="list-style-type: none">■ <code>subAction</code>: the <code>action</code> is used in a subsequent <code>actionList</code> (a menu of actions), or■ <code>actionList</code>: the <code>action</code> has an action list; a menu of actions.
<code>disabled</code>	Disables the action initially.
<code>checkable</code>	Allows the use of the <code>checked</code> attribute.
<code>checked</code>	Makes the <code>action</code> initially checked.
<code>editable</code>	The action is enabled when the cellview is editable.
<code>modified</code>	The action is enabled when the cellview has been modified but not saved.
<code>selected</code>	The action is enabled when the cellview is editable and at least one object is selected.
<code>assistantsPresent</code>	The action is enabled when the workspace contains at least one assistant. The action is enabled even if all the assistants are hidden.
<code>notMultiSheet</code>	The action is enabled when the property <code>schType</code> is not set to <code>index</code> . A schematic with the property <code>schType</code> set to <code>index</code> means that it is the index of a multi-sheet schematic.
<code>tooltip</code>	Specifies tooltips for user-defined toolbars. These tooltips are displayed in the status bar of the application.

Related Topics

[Customize Toolbar Definition Files](#)

[Format for Item Type Typein, Separator, and InheritToolbarsForm](#)

Format for Item Type ComboBox

Use these mandatory and optional fields to define comboBox:

Mandatory Field	Description
name	The name of the comboBox.
tooltip	The text used for the comboBox <i>Tooltip</i> .
items	<p>A list of strings, a string, or a SKILL expression that form the items in the comboBox.</p> <ul style="list-style-type: none">■ 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 <code>nil</code>.
callback	The SKILL string to be called.
width	The width to be given to <code>hiCreateToolbarComboBox</code> .
value	<p>The value of the comboBox, which could be a string or a SKILL expression.</p> <p>If value field is a string, the value of the comboBox is assigned the string as is.</p> <p>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".</p>

Optional Field	Description
invisible	When this field is included and set to <code>t</code> , the comboBox item remains hidden initially.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Optional Field	Description
<code>disabled</code>	When this field is included and set to <code>t</code> , the <code>comboBox</code> remains disabled initially.
<code>editable</code>	The action is enabled when the cellview is editable. Default is <code>t</code> .
<code>prompt</code>	A string that can be used to prompt user action. Default is <code>" "</code> .

Related Topics

[Customize Toolbar Definition Files](#)

[Format for Main Toolbar Definition](#)

[Format for Item Type Action](#)

Format for Item Type `typein`, Separator, and InheritToolbarsForm

Use these mandatory and optional fields to define a `typein`:

Mandatory Field	Description
<code>name</code>	The name of the <code>typein</code> .
<code>tooltip</code>	The text used for the <code>typein</code> <i>Tooltip</i> .
<code>callback</code>	The SKILL string to be called.
<code>value</code>	The value (string) of the <code>typein</code> .
<code>width</code>	The width to be given to the <code>typein</code> .

Optional Field	Description
<code>invisible</code>	When this field is included and set to <code>t</code> , the <code>typein</code> item is hidden initially.
<code>disabled</code>	When this field is included and set to <code>t</code> the <code>typein</code> is disabled initially.
<code>editable</code>	The action is enabled when the cellview is editable. Default is <code>t</code> .
<code>prompt</code>	A string that can be used to prompt user action. Default is <code>" "</code> .

Use these mandatory and optional fields to define a separator:

Mandatory Fields	Description
<code>name</code>	The name of the <code>separator</code> .

Optional Fields	Description
<code>invisible</code>	Hides the <code>separator</code> item initially.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Use this field to define the inheritToolBar:

Mandatory Field	Description
<code>inheritToolbarsFrom</code>	Specifies the string name of an application whose toolbars is inserted in this position in the application of the toolbar.

Related Topics

[Customize Toolbar Definition Files](#)

[Format for Main Toolbar Definition](#)

[Format for Item Type Action](#)

Using and Resetting Toolbars

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 the Virtuoso Design Editor toolbar files.

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 Studio 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.

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`.

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:

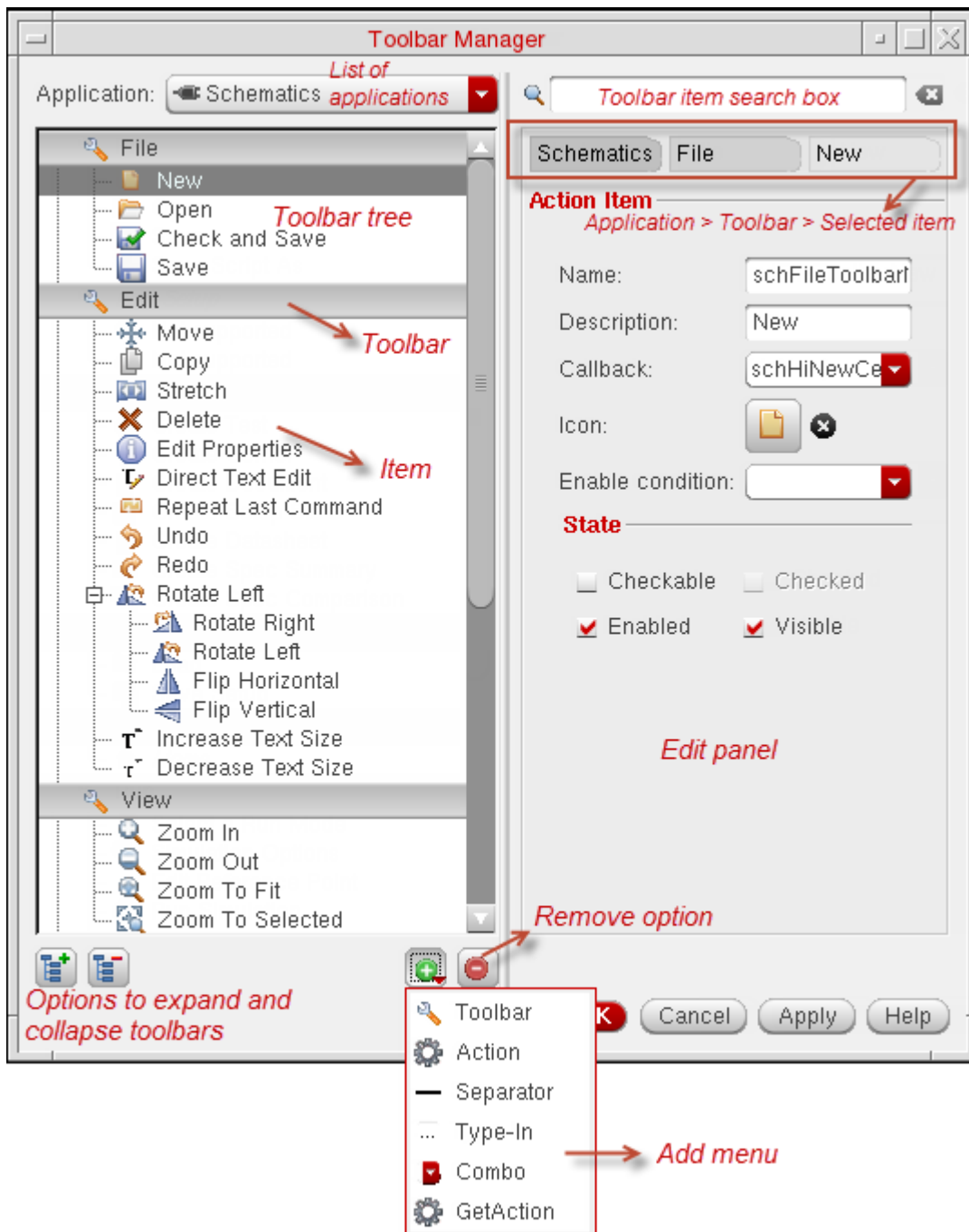
- ☐ Choose *Options – Toolbars* from the Virtuoso CIW.
- ☐ Choose *Window – Toolbars – Customize* from within a Virtuoso session window.
- ☐ Right-click the toolbar in the application and select *Customize*.

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.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization



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.

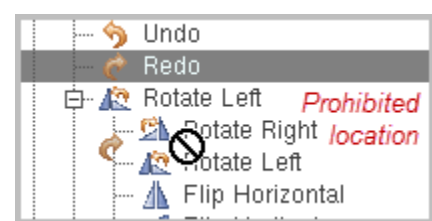
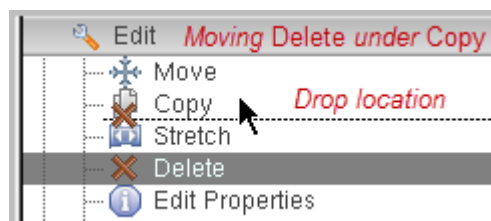
3. Change the application toolbars, as required.

Action	Procedure
Add a new toolbar	<ol style="list-style-type: none">1. Right-click the Toolbar tree and click <i>Add</i> or click <i>Add</i> at the bottom of the form to access the <i>Add</i> menu.2. Click <i>Toolbar</i>.3. Specify the toolbar details. For more information, see Using and Resetting Toolbars.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Action	Procedure
Add a new item	<ol style="list-style-type: none"> 1. Select the toolbar in the Toolbar tree. 2. Click the <i>Add</i> button to access the <i>Add</i> menu. 3. Select the item type. You can choose from: <ul style="list-style-type: none"> <input type="checkbox"/> <i>Action</i>: Adds an action item to perform an action. <input type="checkbox"/> <i>Type-in</i>: Adds an item with an input box. <input type="checkbox"/> <i>Combo</i>: Adds an item with a list for selection. <input type="checkbox"/> <i>Separator</i>: Adds a separator for differentiating between items. <input type="checkbox"/> <i>GetAction</i>: Passes the name of an item to the action retrieval function specified on the item's toolbar. This option allows actions to be reused. <p>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 <i>Enter</i> or <i>Return</i> key is pressed regardless of whether the value in the field has changed, select the <i>Always execute callback when "Enter" key is pressed</i> check box.</p> <ol style="list-style-type: none"> 4. Specify the item details on the Edit panel. For more information, see Toolbar Definition File Format.
Move an item	<p>➔ Select the item on the Toolbar tree, drag it to the new location, and drop the item.</p> <p>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.</p>



Virtuoso Studio Design Environment User Guide

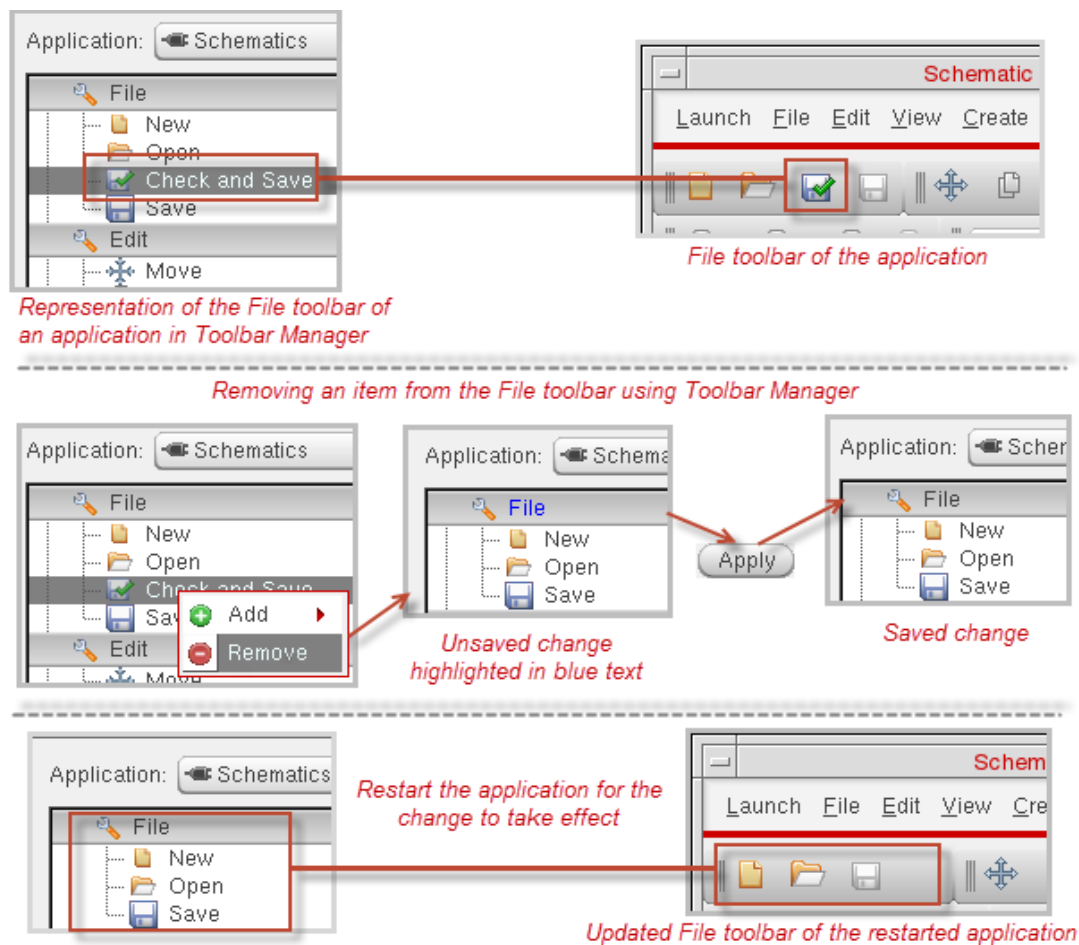
Virtuoso Studio Design Environment Customization

Action	Procedure
Edit a toolbar or item	<ol style="list-style-type: none"> 1. Select the toolbar or item on the Toolbar tree. 2. Edit the details on the Edit panel.
Delete a toolbar or item	<ol style="list-style-type: none"> 1. Select the toolbar or item on the Toolbar tree. 2. Click the <i>Remove</i> 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: These changes 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.





Reset the 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.

Note: This action cannot be undone, make sure that the correct applications are selected.

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.

Related Topics

[.cadence Hierarchy](#)

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 unchecks any others in that group. This is typically implemented via a SKILL callback for each item—based upon which button was pressed, the callback adjusts all buttons to reflect the desired state. In this case, if one is not aware of these effects of the SKILL callback, the behavior seems inconsistent with the behavior specified for those toolbar items.

Related Topics

[Using and Resetting Toolbars](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Customization

Virtuoso 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. 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 .

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.

Related Topics

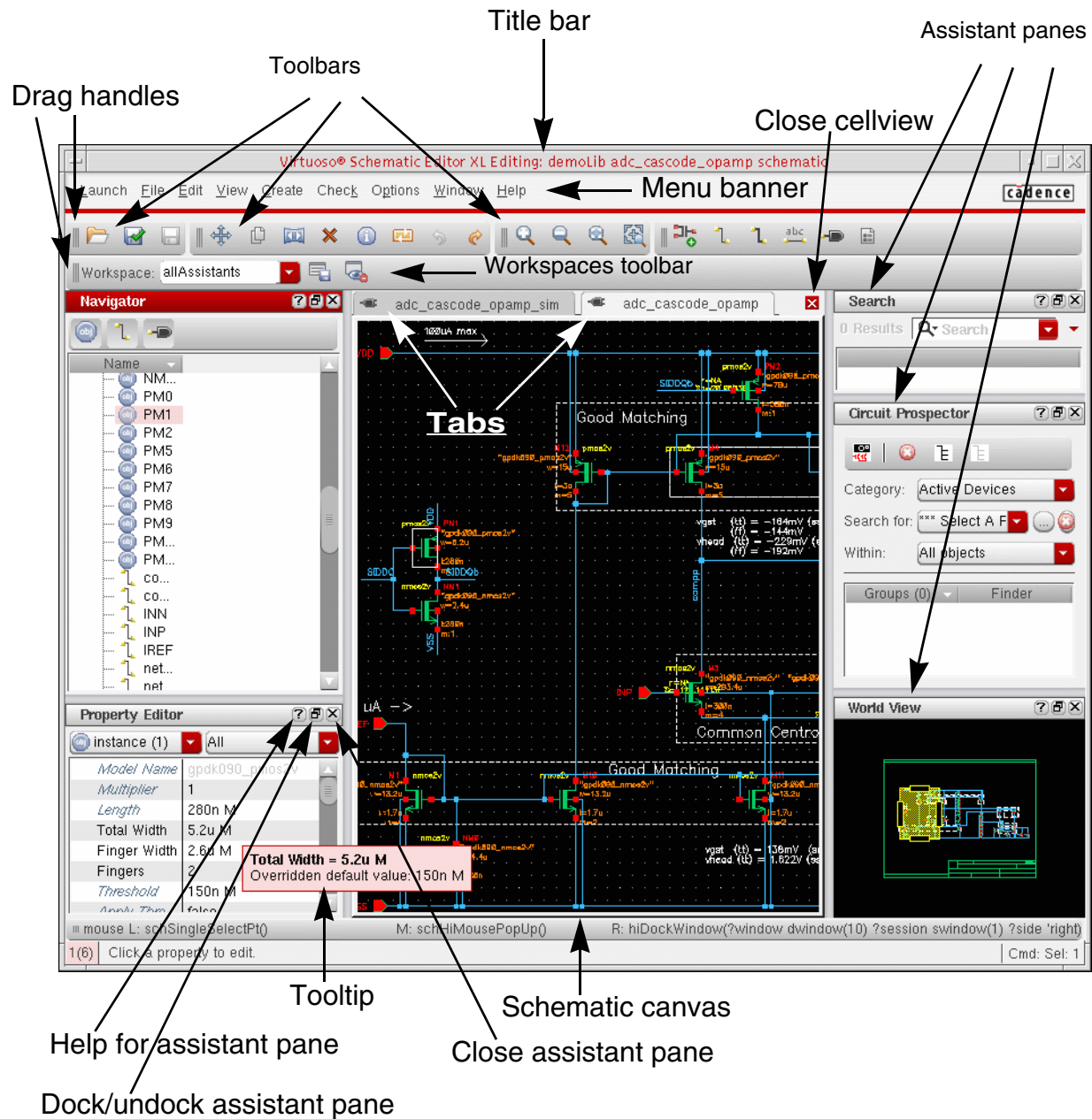
[Workspace Configuration](#)

[Workspace Search Order](#)

[Tabs in a Session Window](#)

Workspace Configuration

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.



You can define the following properties of your window layout in a workspace:

- Docked, floating, and hidden assistant panes

- Docked and hidden toolbars
- 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. 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.

Related Topics

[Session Windows and Workspaces](#)

[Setting the Default Workspace for an Application](#)

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

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 [Cadence Setup Search File: setup.loc](#).

The program uses the first matching workspace definition it finds. Virtuoso Studio Design Environment applications use the following algorithm to determine the set of available workspaces for each application:

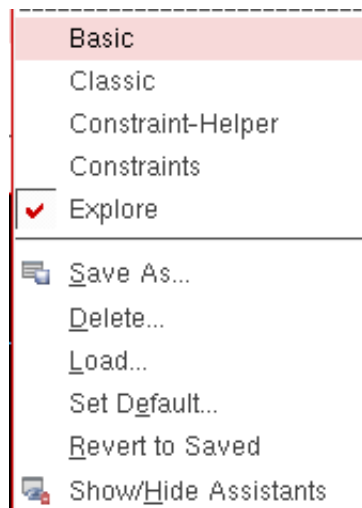
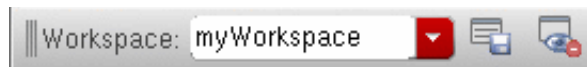
- For each registered tool `$t` in each registered application (such as `adexl-schematic`)
 - For each search path component `$c` in the `setup.loc` file
 - For each matching file `$f` in
`$c/.cadence/dfII/workspaces/$t/*.workspace`
Cache the definition of workspace `$f` for tool `$t`

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces

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/
```

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces

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/
```

Related Topics

Starting Virtuoso Studio

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.

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.

Tabs in a Session Window

You can have one or more cellviews open on one or more tabs in a session window. 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.

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.

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces

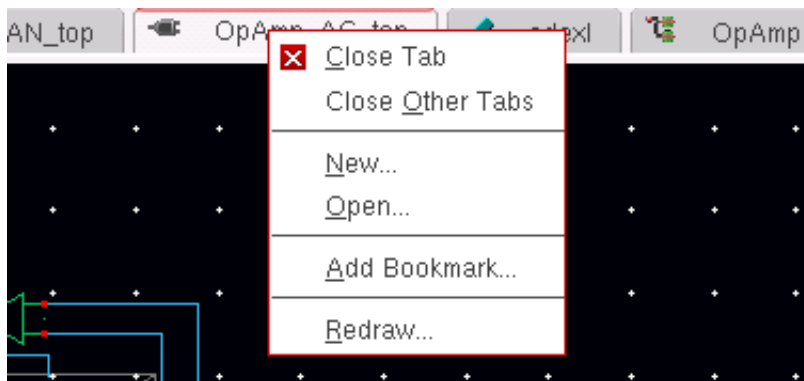
You can have several tabs in your session window, each running a different application. Only one tab is considered to be active at any one time.

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.

You can switch between tabs using the `Ctrl+Tab` keybinding

Tab Pop-Up Menu Options

If you right click over a tab, a pop-up (context) menu gets displayed.



Option	Description
<i>Close Tab</i>	Closes the current tab.
<i>Close Other Tabs</i>	Closes all other open tabs except for the current tab.
<i>New</i>	Displays the New File Form.
<i>Open</i>	Displays the Open Cell form.
<i>Add Bookmark</i>	Displays the Add Bookmark form.
<i>Redraw</i>	Refreshes the current design canvas window.

Related Topics

[Opening a Cellview from the CIW](#)

[Closing Cellviews in Virtuoso](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces

[Creating a New Cellview](#)

[Bookmarking Designs](#)

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.

Related Topics

[Tabs in a Session Window](#)

[Setting the Default Workspace for an Application](#)

Workspace Features

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.

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.

Related Topics

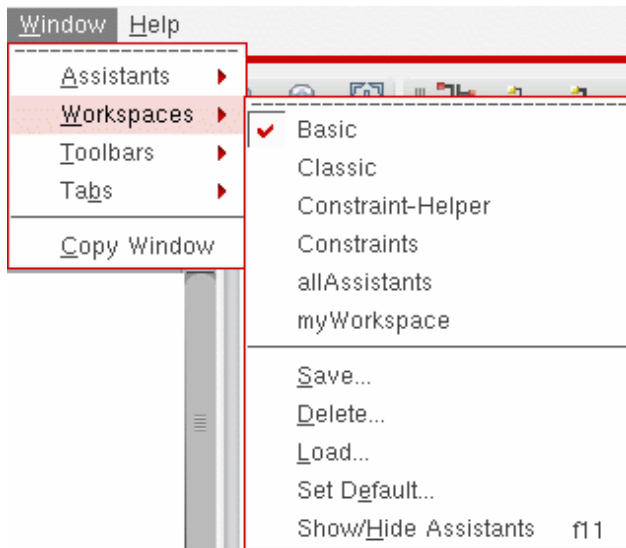
[Customizing a Cadence Workspace](#)

Selecting a Workspace

To select a workspace using the menu banner:

1. Choose *Window – Workspaces*.

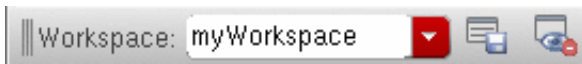
A submenu of workspaces appears listing only those configurations available for use with the current cellview/application.



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.

Related Topics

[Saving a Workspace](#)

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.

To add an assistant pane or toolbar to your custom workspace using the right-click menu:

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.

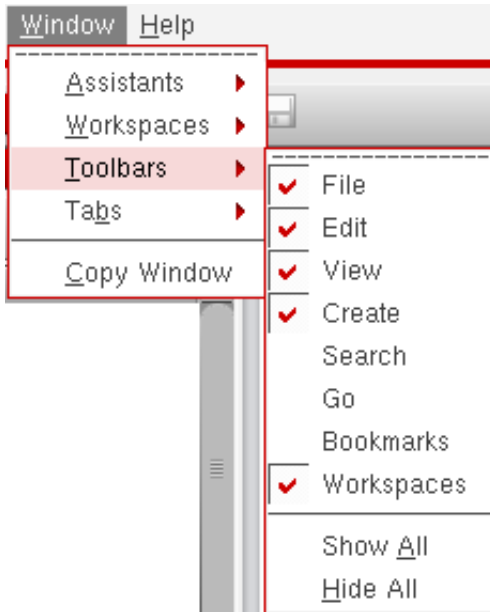


2. Select an item that is not currently active which mean the item does not display a check mark to the left of its name.

The item appears in your workspace. If you display the right-click menu again, you see a check mark next to the item. To add a toolbar to your custom workspace using the *Window* menu:

3. 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.

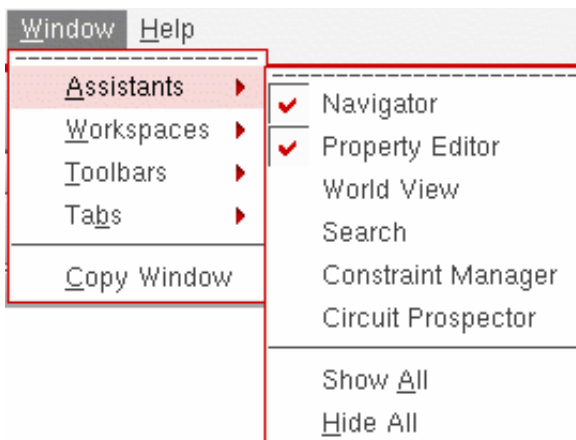


4. Select a toolbar name that is not currently active.

To add an assistant pane to your custom workspace using the *Window* menu:

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.



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. A check mark appears next to the pane name If you display the submenu again.

Related Topics

[Controlling Assistant Panes](#)

[Modifying the Size and Location of an Assistant Pane](#)

Removing Items from a Workspace

You can remove assistant panes and toolbars from your workspace using the *Window* menu or a right-click menu.

To remove an assistant pane or a toolbar from your custom workspace using the right-click menu:

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.

To remove a toolbar from your custom workspace using the *Window* menu:

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.

To remove an assistant pane from your custom workspace using the *Window* menu:

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.

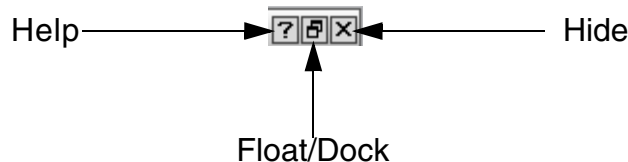
Related Topics

[Workspace Features](#)

[Selecting a Workspace](#)

Controlling Assistant Panes

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* buttons.



To make a docked assistant pane float:

1. In the upper right corner of the docked pane, click the Float button.
2. In the title bar of the docked pane, double-click.
3. Drag a docked pane by its title bar to an open area and release the mouse button.
4. The pane is now floating.

To dock a floating assistant pane:

1. In the upper right corner of the floating pane, click the Dock button.
2. In the title bar of the floating pane, double-click.
3. Drag the floating pane close to any edge of your session window until you see its outline where you want it and drop it.
4. The pane is now docked.

For more information, see [Modifying the Size and Location of an Assistant Pane](#).

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.

You can click the *Float* button to detach an assistant from an existing group of tabbed assistants. Alternatively, you can drag the assistant away from the grouped assistants.

Hiding Assistant Panes

To hide an assistant pane:

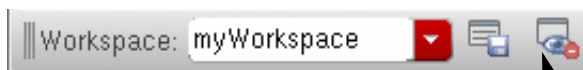
- ➡ Click the Hide (x) button in the upper right corner of the pane.

The program removes the pane from your workspace.

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](#).

To hide all assistant panes with a single action:

- ➡ On the far right end of the Workspace toolbar, click the Show/Hide Assistants button.



Show/Hide Assistants

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.

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*.

To toggle assistant panes off:

- ➡ 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.

Ctrl+F11 can also be used to toggle all toolbar display, and Shift+F11 toggles both toolbar and assistant display.

To get help on an assistant pane:

- ➡ In the upper right corner of the pane, click the question mark icon.



The Help page for the assistant appears in the Doc Assistant window.

Related Topics

[Workspace Features](#)

[Loading and Deleting a Custom Workspace](#)

Modifying the Size and Location of an Assistant Pane

In this topic, learn to 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 or Floating Assistant Pane

To modify the size of a docked or floating assistant pane:

- ➔ Click-drag any edge or corner.

You can identify an edge that you can move when the mouse cursor appears as two parallel lines with outward-facing arrows:



Your assistant pane appears taller or shorter, wider or narrower, according to the edge is repositioned.

Changing the Location of a Docked Assistant Pane

To change the location of a docked assistant pane:

- ➔ 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.

Changing the Location of a Floating Assistant Pane

To change the location of a floating assistant pane:

- ➔ Drag-and-drop the pane by its title bar to a new location.

If you drag the floating assistant pane to a location within your session window, the pane becomes 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.

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.

Modifying the Length or Location of a Toolbar

To modify the length of a horizontally-oriented toolbar:

- ➡ Click-drag its handle to the left or right.

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:

- ➡ 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:

- ➡ 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.

Related Topics

[Saving a Workspace](#)

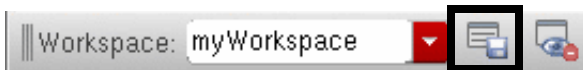
Saving a Workspace

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.

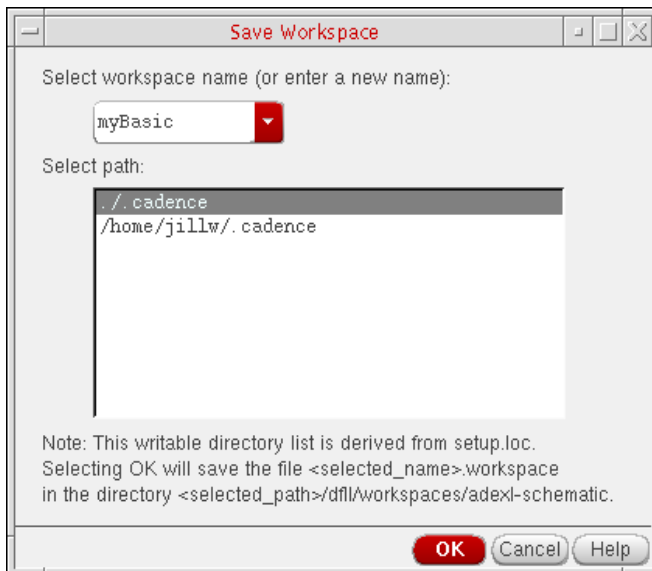
If you save a new version of a default workspace, you might not be able to retrieve the original.

To save your workspace:

1. On the Workspace toolbar, click the Save Workspace icon.



The Save Workspace form appears.

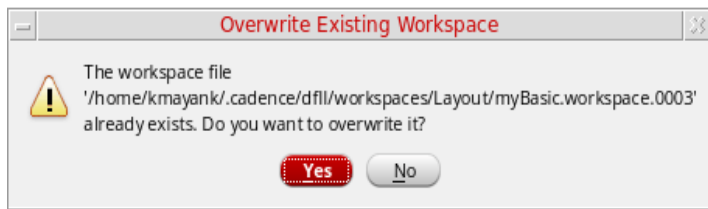


2. In the *Select workspace name* drop-down combo box, either select a workspace name or type a new name.
3. Select the *Overwrite existing workspace* check box if you want to overwrite the existing workspace.

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.

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces



Click *Yes* to continue.

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"
```

Related Topics

[Workspace Search Order](#)

[Loading and Deleting a Custom Workspace](#)

Loading and Deleting 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:

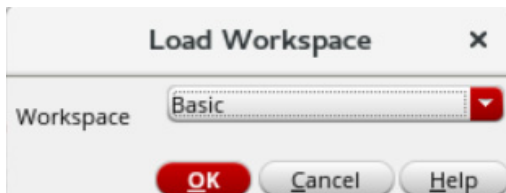
- ➔ In the *Workspace* drop-down combo box, select a workspace.

The program configures your session window using the workspace you selected.

To load a custom workspace using the *Window* menu:

1. Choose *Window – Workspaces – Load*.

The Load Workspace form appears.



2. In the *Workspace* drop-down combo box, select a workspace.
3. Click *OK*.

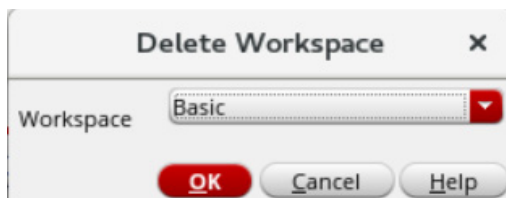
The program configures your session window using the workspace you selected.

Deleting a Custom Workspace

To delete a custom workspace:

1. Choose *Window – Workspaces – Delete*.

The Delete Worksapce form appears.



2. In the *Workspace* drop-down combo box, select the workspace you want to delete.
3. Click *OK*.

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces

The program deletes the workspace. If you delete the current workspace, the program displays the default workspace.

Related Topics

[Saving a Workspace](#)

[Load Workspace Form](#)

Customizing a Cadence Workspace

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:

1. Select a Cadence workspace.
2. Modify the Cadence workspace to your specifications.
3. Save your custom workspace.

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. 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 finds 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 finds the Cadence workspace first.

Reverting to a Saved Workspace

To revert to the last version of the current workspace saved to disk:

- ➔ Choose *Window – Workspaces – Revert to Saved*.

The last-saved version of the current workspace appears in the session window.

Related Topics

Selecting a Workspace

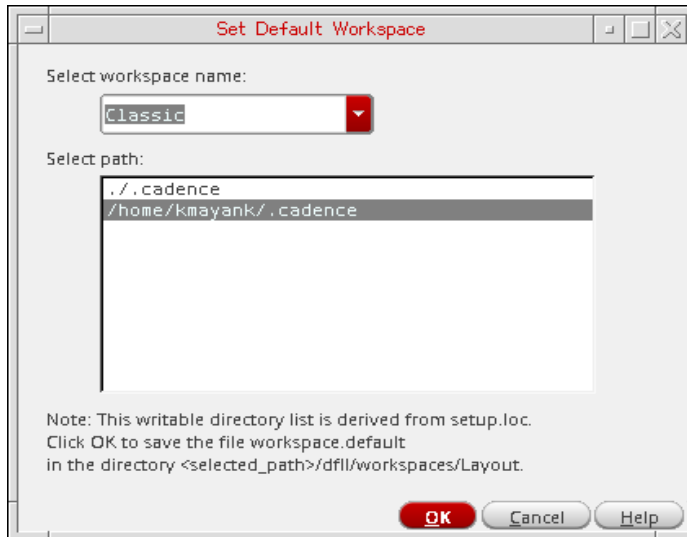
Workspace Search Order

Setting the Default Workspace for an Application

To select a current workspace as the default for a particular application:

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 appears 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) gets listed.

Note: Assuming that your home directory has been set up as a member of the CSF, the workspace gets 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.

Related Topics

Selecting a Workspace

Virtuoso Studio Design Environment User Guide

Virtuoso Workspaces

Design Editor Plugins

A plugin is an application that adds extra capability to a main application, or to another plugin. A plugin that is implemented with the design editor (DE) plugin infrastructure is called a DE plugin. The DE plugin is not a DE subapplication. Infact, it is an add-on to a DE main application, such as schematic or layout.

Two major advantages of a DE plugin over a subapplication are as follows:

- Runs without replacing the main application
- Co-exists with other plugins that are not being replaced

However, a limitation of a DE plugin is that you cannot register a plugin with an `appName` and/or a `className` that has already been taken.

You can create a DE plugin by defining a subclass of the DE base class, `DEPlugin` and re-implementing certain public methods of the class that can be *overridden*, if required.

The public methods that can be *overridden* are named with the prefix `deo`, where the character `o` indicates *override*. You should not reimplement methods that are not prefixed with `deo`.

For more information about defining a plugin subclass and the three main types of public methods that can be *overridden* in the DE Plugin subclass, see [Registering DE Plugins](#).

Features of a Design Editor Plugin

A DE plugin has the following main features:

- Can own menus, toolbars, assistants, and bindkeys
- Can customize banner menus and toolbars of other applications
- Can co-exist with multiple plugins
- Does not replace the running application when installed

- Does not have its own workspace

You can also create and use your own DE plugins. To be able to do this, you need to create, register, and then install the DE plugin.

Defining and Customizing New Toolbars

You can choose to create new toolbars for your plugin or customize existing toolbars of a main application, or those of another plugin.

1. Define a new toolbar for your plugin by providing a toolbar definition file for the plugin in the same way as you would provide for a DE main application.
2. Inherit toolbars from super plugin class by using the `inheritToolbarsFrom` option of the toolbar definition file.

When installing a plugin, DE loads the toolbars specified by the toolbar file of the plugin, including those inherited through the `inheritToolbarsFrom` option. DE does not read the toolbar file of its super-plugin-class(es), unless it happens to be included by the `inheritToolbarsFrom` option.

Support Bindkeys in DE Plugin

To support bindkeys for your plugin, DE calls `hiSetBindkeys` by using the value returned by the `deoGetBindKeys` method when a plugin is installed.

The returned list should append more specific key-bindings after those inherited from super-application(s). This is because if the same `t_key` appears more than once in its argument list, `hiSetBindKeys`, uses the last key-binding definition to override the one(s) defined previously.

Banner Menus

You can create a plugin for customizing the banner menu of an application window.

When installing a plugin, DE calls the `deoMenuTrigger` defined by the plugin. The menu trigger should not install the menus itself; this is done by DE.

The menu trigger merely returns a list of the menus that are to be installed. The menu trigger is passed trigger arguments to customize the menu list according to data specified as trigger arguments.

Workspace Defined by DE Plugin

A workspace is a property of a DE main application. A plugin does not have its own workspace. However, a main application workspace can contain assistants and toolbars defined by a plugin. When saving a workspace of a main application to a workspace file, the workspace includes information of all plugins active at that time.

When the saved workspace is re-applied later, all plugins contained in the workspace are reinstalled. If a plugin is not defined at that time, a warning message is issued.

When a plugin, which has either assistant or toolbar, is installed on top of a main application, the window is considered to be a different application. The workspace applied to this window can be different from the workspace applied to another window that has the same main application but no plugin or different plugin(s) installed.

Related Topics

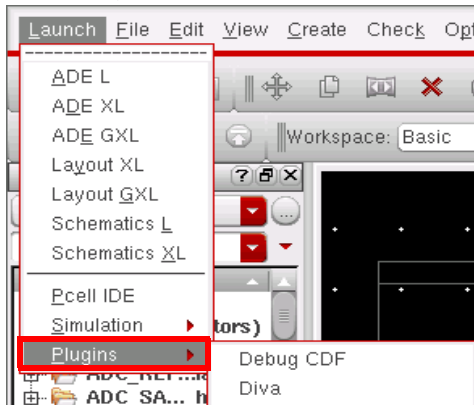
[Accessing a Design Editor Plugin](#)

Accessing a Design Editor Plugin

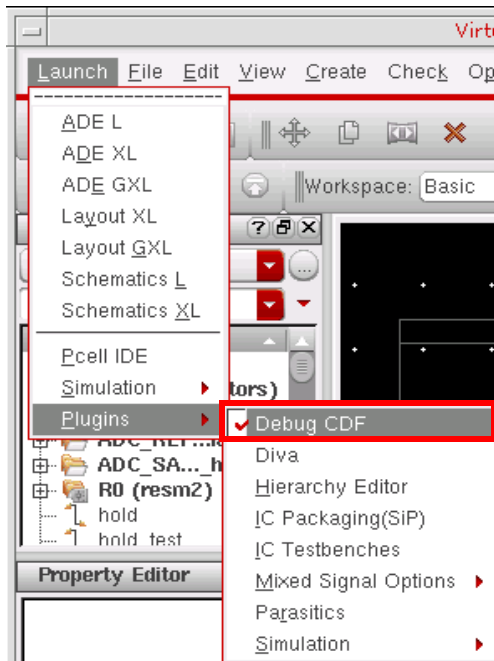
To access existing Virtuoso DE plugins from the schematic or layout windows:

1. Choose *Launch – Plugins*.

The Plugins submenu is displayed.



2. From the Plugins submenu, choose the name of the plugin you want to install.
When the plugin is installed, the plugin is displayed and its name appears with a check mark in the submenu as shown in the figure below.



To remove an installed plugin from the current window, select the active plugin menu item. The plugin is removed. Choose *Launch – Plugins* to see that the check mark no longer appears next to the Plugins submenu item.

Related Topics

[Registering a Design Editor Plugin](#)

[Installing and Removing a Design Editor Plugin](#)

Registering a Design Editor Plugin

To make a plugin available for installation on an application window:

1. Register the plugin by using `deRegPlugin`. Specify the plugin name (`appName`) and the class and superclass names (`className` and `superClass`) which implement the plugin.
2. After a plugin is registered, use the `appName` and `className` to uniquely identify the plugin. For example, to register `My Plugin` so that it can be installed on all applications associated with view type `schematic`, such as VSE and VSE XL applications, use the following code:

```
deRegPlugin(  
    "My Plugin"  
    'MyPlugin  
    'DEPlugin  
    ?appTitle "My &Plugin"  
    ?pluggableViewTypes '("schematic")  
)
```

3. Retrieve a copy of the registration information of a plugin by defining `deGetAppClassInfo`. This function returns a copy of the registration information as a Disembodied Property List (DPL) associated with an application. For example, the following code retrieves a copy of the registration information associated with `My Plugin`:

```
(deGetAppClassInfo "My Plugin")  
  
=>  
(nil appType plugin  
appName "My Plugin"  
className MyPlugin  
superClass DEPlugin  
bindkeys nil  
""  
pluggableMainApps nil  
pluggableViewTypes '("schematic")  
shortDescription ""  
)
```

However, a limitation with `deRegPlugin` is that the function issues a warning if:

- A class symbol is already used by an application.
- A plugin application name is already registered, or used as an application tool name, such as "&ADE L".

Related Topics

Converting an Existing Application to a Plugin

Installing and Removing a Design Editor Plugin

After you have registered the plugin, you can install or activate the plugin:

- ➔ Use `deInstallPlugin` in an application window. The application window in which a plugin is to be installed should be a tab window.

For each session window, DE makes only one instance of a plugin. This means that if an instance of the plugin already exists in a tab window, `deInstallPlugin` returns the instance. And, if there exists an instance of a plugin in another tab of the same session window, `deInstallPlugin` uses the instance.

Note: If not already loaded, the class and superclass (from which this plugin is implemented) are auto-loaded when `deInstallPlugin` is applied to the `appName` or `className` if the `autoloadClass` property is defined on the `className`.

Once a plugin is installed in an application window, the plugin stays active as long as it is pluggable to the application. For example, a plugin, which is pluggable to `schematic` view type, remains active regardless of whether you launch a new schematic application in the window or open another schematic cellview in the same window.

However, `deInstallPlugin` issues a warning if any or all of the following conditions are true:

- The plugin is already installed in the given window.
- The given window is not a main design window.
- The plugin is not pluggable to the application running in the given window.
- The arguments are invalid.

To remove a plugin from a given application window:

- ➔ Call `deRemovePlugin` in an application window. You can specify either the plugin name, or the class name from which this plugin is implemented.

The function returns `t` if the plugin is successfully removed from the specified application window. It returns `nil` if either the named plugin is not found in the specified window, its exit trigger has returned `nil`, or the arguments are invalid. If any of these conditions is true, Virtuoso does not remove the plugin and issues a warning.

Note: If the window argument given to `deRemovePlugin` is a session window, the plugin is removed from all tab windows associated with the session window.

Related Topics

[deInstallPlugin](#)

[deRemovePlugin](#)

[Registering a Design Editor Plugin](#)

[Auto loading Your Classes](#)

Converting an Existing Application to a Plugin

You can convert an existing DE subapplication to become a plugin. The minimum steps that you need to perform to convert an existing sub-application to a plugin are as follows:

1. Register the subapplication as a plugin using the same `appName` and `appTitle`.

You can execute existing scripts that install the subapplication with `deInstallApp` function because `deInstallApp` is updated such that if its second argument (`t_appName`) is being registered as a plugin with `deRegPlugin`, then `deInstallPlugin` is called to perform the installation.

2. Re-implement enable trigger, menu trigger, post install trigger, and/or exit trigger, if needed. In many cases, existing trigger code can be reused by wrapping it in a `deo` method.

For application that opens its own window, it is important to redefine the exit trigger because when you select the plugin on the submenu to remove it, the plugin gets a chance to close the window. In addition, the window closing procedure should call `deRemovePlugin` to remove itself from the design window.

Candidates of sub-applications that can be converted to a plugin include DE sub-applications that define the enable, menu, post-install triggers, and/or the exit trigger.

Related Topics

[Registering a Design Editor Plugin](#)

Virtuoso Studio Design Environment User Guide

Design Editor Plugins

Bookmarks and Views in Virtuoso Studio Design Environment

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

You can also access bookmark functionality from the CIW (*File – Bookmarks*). When opening a bookmark from the CIW, the design gets 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](#).

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.

Related Topics

[Bookmarking Designs](#)

[Restoring a Bookmark](#)

[Managing Bookmarks](#)

[Managing Composite Bookmarks](#)

Bookmarking Designs

To add the Bookmarks toolbar to your current session window:

- 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*.

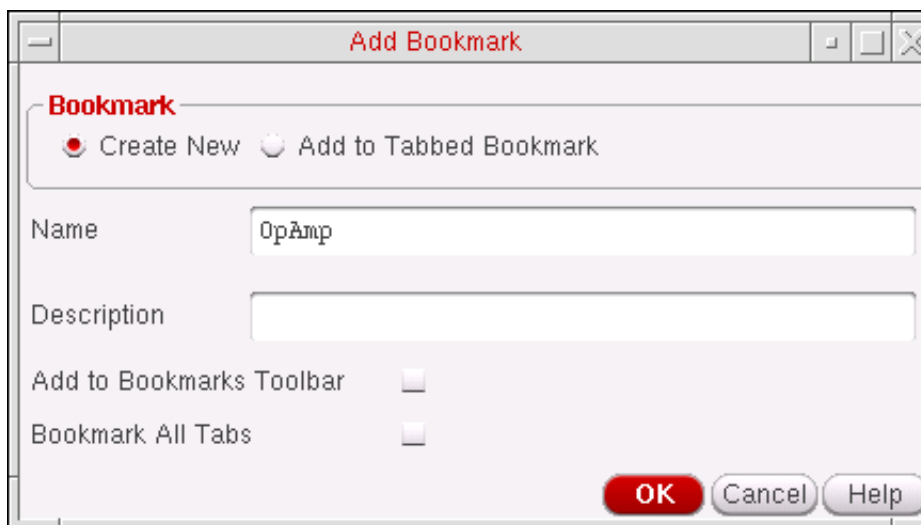
The program adds the Bookmarks toolbar to your current session window.

Note: Other session windows are not affected by this addition. You can bookmark any number of cellviews.

To add a bookmark for the current design or a collection of open designs:

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. You can also right-click over any existing session window tab, and select *Add Bookmark* to display the Add Bookmark form.

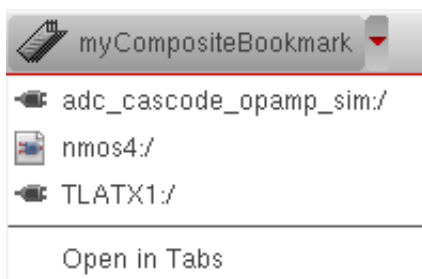
2. Select the *Create New* bookmark option.
3. (Optional) In the *Name* field, type a different name for your bookmark.
4. (Optional) In the *Description* field, type a description for your bookmark.

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.

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.

7. Click *OK*

The bookmark gets created and optionally added to the *Bookmarks* toolbar. Otherwise, the bookmark gets 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.

Related Topics

[Adding Assistant Panes and Toolbars to a Workspace](#)

[Adding Bookmarks to Existing Composite Bookmarks](#)

[Restoring a Bookmark](#)

[Managing Bookmarks](#)

[Add Bookmark Form](#)

Restoring a Bookmark

You can restore single or composite bookmarks using:

- ➔ *File – Bookmarks* menu or from the Bookmarks toolbar.

A composite bookmark is a bookmark created from all tabs in a session window. 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:



You can also access bookmark functionality from the CIW (*File – Bookmarks*). When opening/restoring a bookmark from the CIW, the design gets opened in a new session window rather than opening it as a new tab in the current session window.

Restoring a Single Bookmark

To restore a single bookmark from the Bookmarks toolbar:

- ➔ 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.

To restore a single bookmark using the *File* menu:

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.

Restoring a Composite Bookmark

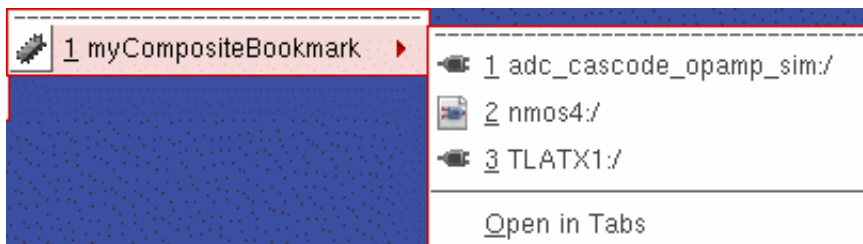
When you restore a composite bookmark, you can choose to restore the entire set of views that make up the composite bookmark, or just one view from the composite.

To restore all views that comprise a composite bookmark from the Bookmarks toolbar:

1. Click the down arrow to the right of the composite bookmark name.
A drop-down menu appears.
If you click directly on the composite bookmark name only the first view listed in the composite bookmark gets opened in a new tab.
2. Select *Open in Tabs*.
Each view in the composite bookmark opens in an individual tab in your session window.
If you do not see your composite bookmark on the Bookmarks toolbar, you might find it on the *File – Bookmarks* menu instead.

To restore a composite bookmark using the *File* menu:

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.

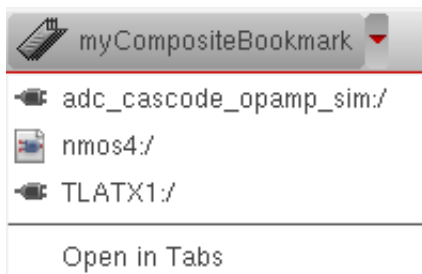


4. At the bottom of this menu, select *Open in Tabs*.
The bookmarked views appear on new tabs in your session window.

Restoring a View from a Composite Bookmark

To restore one view from a composite bookmark on the Bookmarks toolbar:

1. Click the down arrow to the right of the composite bookmark name.
A drop-down menu appears.



If you click directly on the composite bookmark name only the first view listed in the composite bookmark gets opened in a new tab.

2. Select the view you want to restore.

The bookmarked view appears in your session window.

If you do not see your composite bookmark on the Bookmarks toolbar, you might find it on the *File – Bookmarks* menu instead.

To restore a cellview from a composite bookmark using the *File* menu:

1. In either the session window or the Command Interpreter Window, choose *File – Bookmarks*.

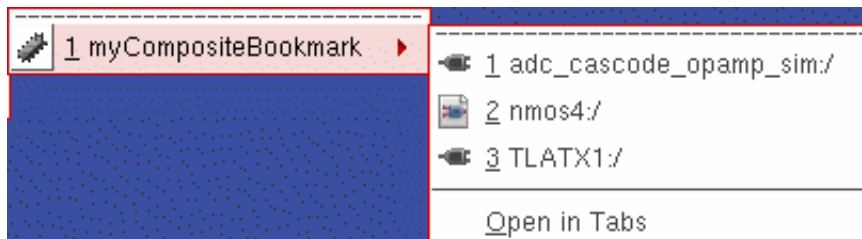
A submenu appears.

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.

Related Topics

[Bookmarking Designs](#)

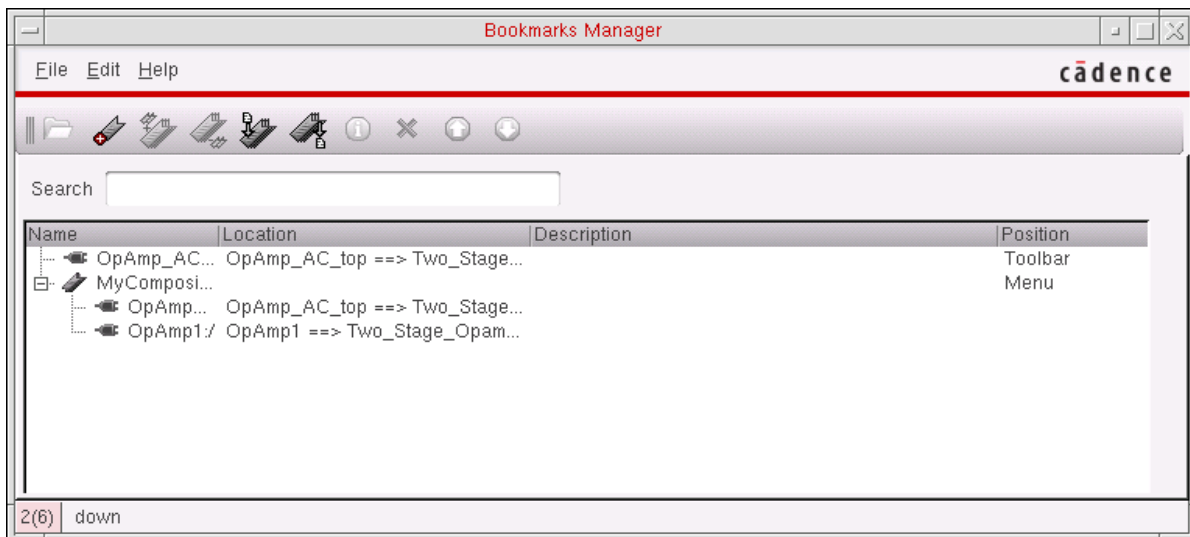
[Managing Bookmarks](#)

Managing Bookmarks

To begin managing your bookmarks:

- ➔ 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	Description
<i>Name</i>	Bookmark name
<i>Location</i>	Cell name following by its <i>lib__cell__view</i> designation
<i>Description</i>	Description of the Bookmark
<i>Position</i>	<i>Toolbar</i> if the bookmark appears on the Bookmarks toolbar; otherwise, <i>Menu</i>

Adding a Bookmark

To add a bookmark:

- ➔ In the Bookmarks Manager window, choose *Edit – Add*.

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.

Importing Bookmarks

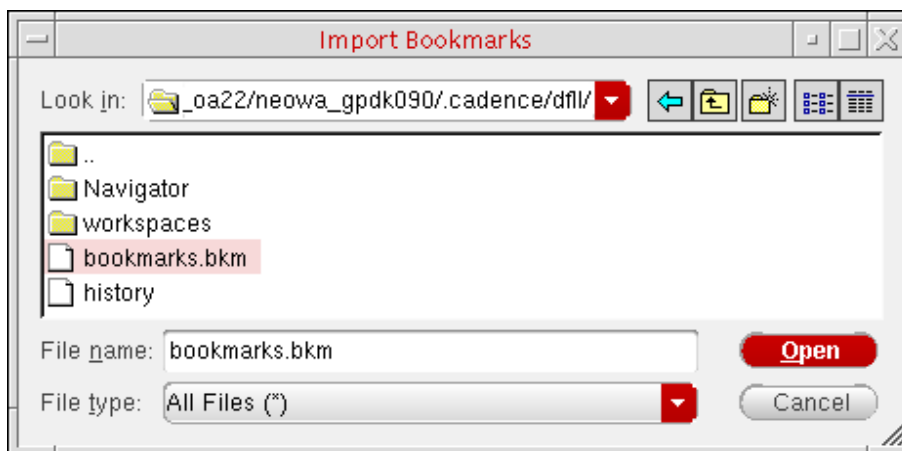
To import a set of bookmarks from a valid bookmark file:

1. In the Bookmarks Manager window, choose *File – Import*.

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.

Exporting Bookmarks

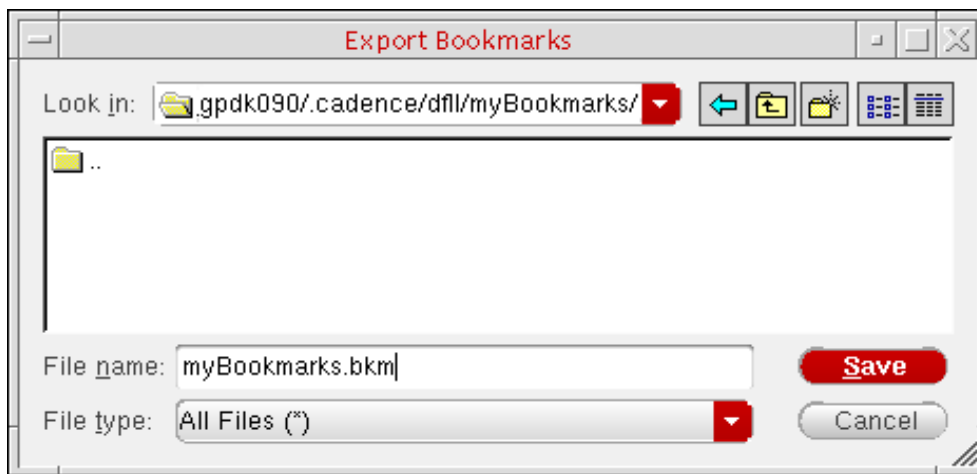
To export your bookmarks to a file:

1. In the Bookmarks Manager window, choose *File – Export*.

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:

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 only be able to re-position them within the composite (collective) bookmark.

Deleting Bookmarks

To delete one or more bookmarks:

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*.

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:

1. In the Bookmarks Manager window, choose *Edit – Properties*.

Alternatively, you can click the Edit Bookmark Properties icon on the toolbar:



The Edit Bookmark Properties form appears.

A screenshot of a dialog box titled "Edit Bookmark Properties". It contains three input fields: "Name" with the text "adc_cascode_opamp_sim", "Description" with the text "myDescription", and "Add to Bookmarks Toolbar" with a checked checkbox. At the bottom right, there are three buttons: "OK" (highlighted in red), "Cancel", and "Help".

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:

1. In the Bookmarks Manager window, select one or more bookmarks.
2. Choose *File – Open*.

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:

Name	Description	Location
<ul style="list-style-type: none"> [-] follower:diffAmp <ul style="list-style-type: none"> [-] follower:/ [-] diffAmp/ 		follower ==> overview.follower:schematic diffAmp ==> overview.diffAmp:schematic

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:

- ➔ 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.

Related Topics

[Bookmarking Designs](#)

[importDuplicateBookmarks](#)

Managing Composite Bookmarks

You can manage existing composite bookmarks with the listed bookmark features.

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*.

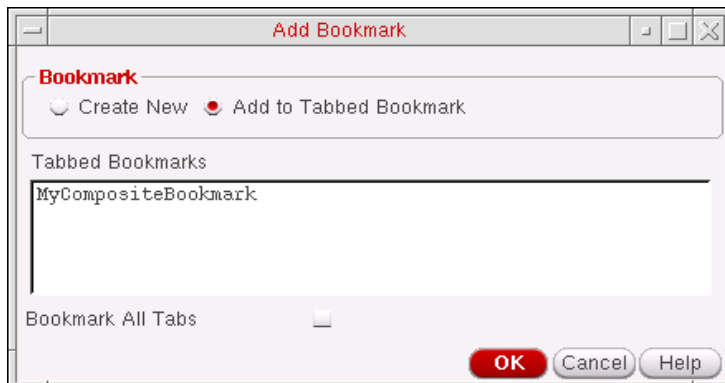
The Add Bookmark form appears.

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.

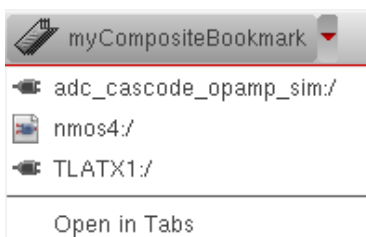
2. Select the *Add to Tabbed Bookmark* option.

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 gets 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*).
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.

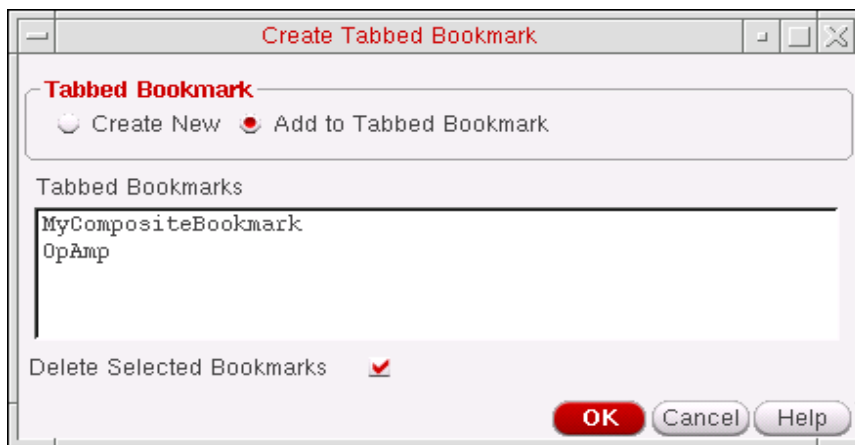
This text appears in the *Description* column of the *Bookmarks Manager* window.

6. (Optional) If you want your composite bookmark to also appear on the *Bookmarks* toolbar, check the *Add to Bookmarks Toolbar* check box.

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.

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 gets created, or the selected bookmarks gets added to an existing tabbed bookmark.

Detaching Bookmarks from Existing Composite Bookmarks

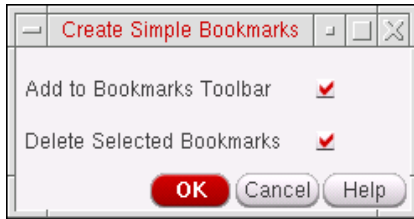
To create simple single bookmarks by either removing them or copying them from existing tabbed composite bookmarks:

1. In the *Bookmarks Manager*, select the bookmarks that you want to detach from an existing composite bookmark.

By detaching a bookmark from a composite bookmark you are not automatically deleting the bookmark from its composite host.

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 also appear in the *Bookmark* toolbar and/or be deleted from the composite it was initially found in.

Related Topics

[Bookmarking Designs](#)

Saving and Restoring Views

To save a view:

1. Choose *View – Save/Restore – Save View*.

The Save View form appears.



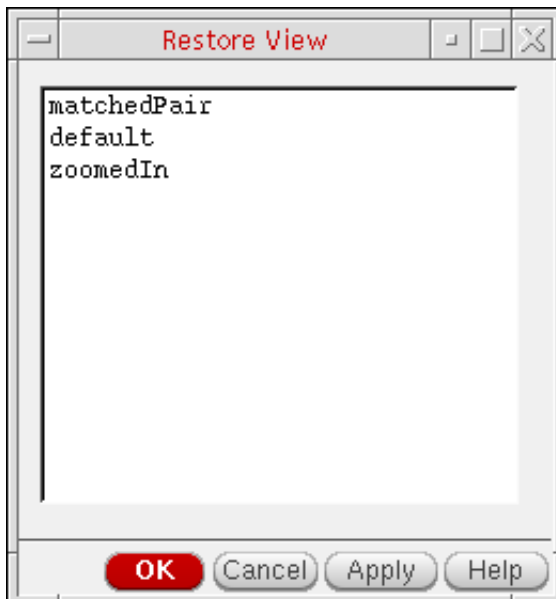
2. In the Name field, type a name for your view.
3. Click *OK*.

Restoring a Saved View

To restore a saved view:

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:

- ➔ Choose *View – Save/Restore – Previous View*.

The program returns to the previous view.

Restoring the Next View

To restore the next view:

- ➔ Choose *View – Save/Restore – Next View*.

The program displays the next view.

Related Topics

[Restoring a Bookmark](#)

[Save View Form](#)

Navigating Cellviews and Hierarchies

You can use the Go toolbar to perform the following operations:

- 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](#).

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

Related Topics

[Accessing the Go Toolbar](#)

[Moving Back through a Design](#)

Virtuoso Studio Design Environment User Guide

Navigating Cellviews and Hierarchies

[Moving Forward through a Design](#)

[Moving Up and to the Top of a Design](#)

Accessing the Go Toolbar

To access the Go toolbar:

1. Select *Window – Toolbars – Go*.
2. 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	Description
<i>Back</i>	Shifts one level back to the previously displayed cell/cellview. The <i>Back</i> drop-down icon (red triangle) displays a list of cell/cellviews that you can choose to go back to. The equivalent SKILL command is <u>deBack</u> .
<i>Forward</i>	Shifts one level forward to the cell/cellview that was displayed prior to the use of the last <i>Back</i> command. The <i>Forward</i> drop-down icon (red triangle) displays a list of cell/cellviews that you can choose to go forward to. The equivalent SKILL command is <u>deForward</u> .
<i>Up</i>	Activates only if you have descended into a hierarchy. Selection of the <i>Up</i> icon will move you up one level in the hierarchy.
<i>Top</i>	Activates only if you have descended into a hierarchy. Selection of the <i>Top</i> icon will move you up to the cell/cellview that represents the top of the hierarchy.

Related Topics

[Moving Back through a Design](#)

[Moving Forward through a Design](#)

[Moving Up and 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 to display a list of previously displayed cells/cellviews.



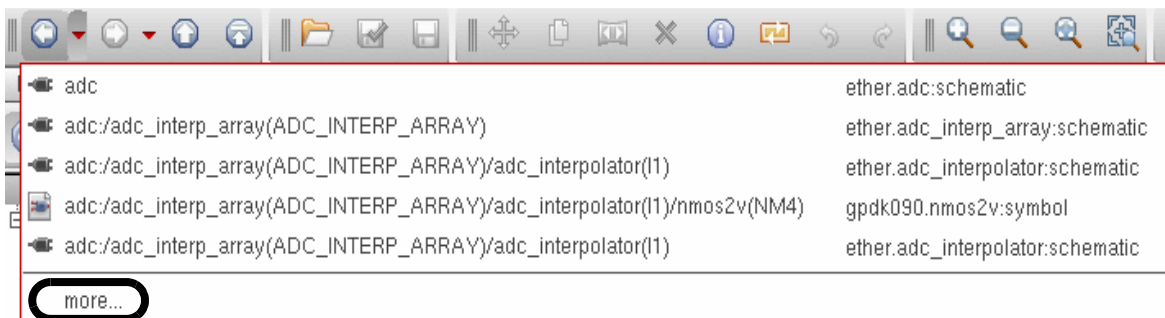
The list presented allows you to select a view and navigate straight to it without having to repeatedly scroll back. The cellview at the top of the drop-down list represents the most recently viewed cellview.

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:

1. On the Go toolbar, click the down-facing red arrow to the right of the Back button.

A drop-down menu appears.

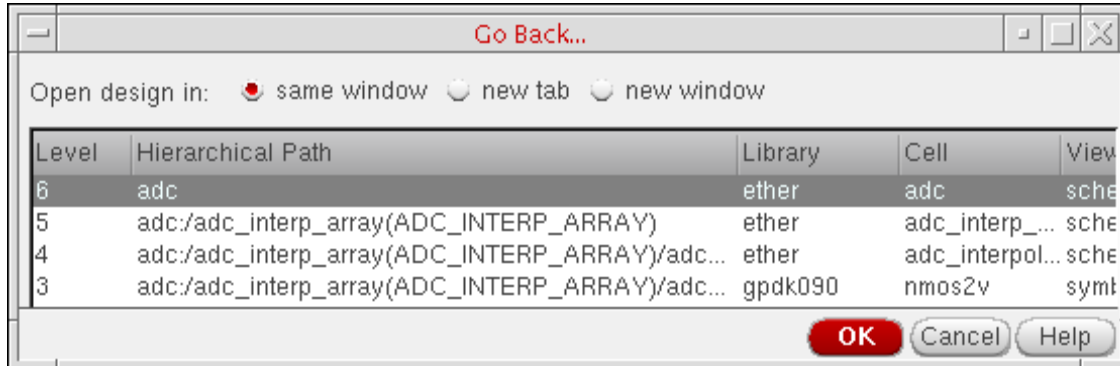


2. Click *more*.

Virtuoso Studio Design Environment User Guide

Navigating Cellviews and Hierarchies

The Go Back form appears.



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.

Related Topics

[Moving Forward through a Design](#)

[Moving Up and to the Top of a Design](#)

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.

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 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.

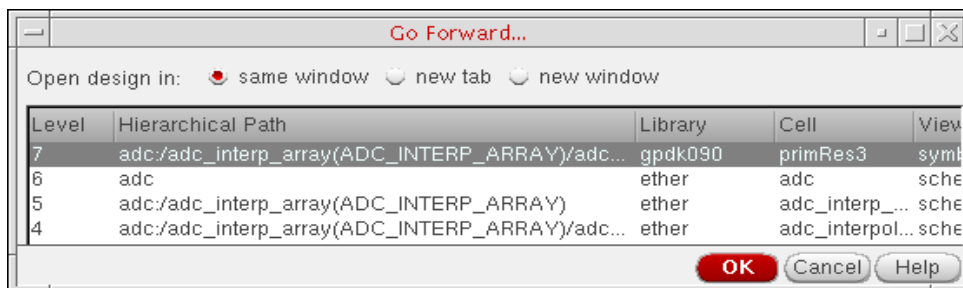
To move forward through the extended hierarchy:

1. On the Go toolbar, click the drop-down icon to the right of the Forward button.

A drop-down menu appears.

2. Click *more*.

The Go Forward form appears.



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.

Virtuoso Studio Design Environment User Guide

Navigating Cellviews and Hierarchies

Related Topics

[Moving Back through a Design](#)

[Moving Up and to the Top of a Design](#)

Moving Up and to the Top of 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.

There is no *Down* icon because you can use the *Back* icon to revisit cellviews.

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.

Related Topics

[Moving Back through a Design](#)

[Moving Forward through a Design](#)

Text Window Options

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 operations in a text window:

- Save a text file
- Use a specific text to search for a particular file
- Print and Refresh a file

To turn off the appearance of the What's New text window every time you start Cadence software, see [Saving Changes in Virtuoso Studio Design Environment](#).

Related Topics

[Saving a Text File](#)

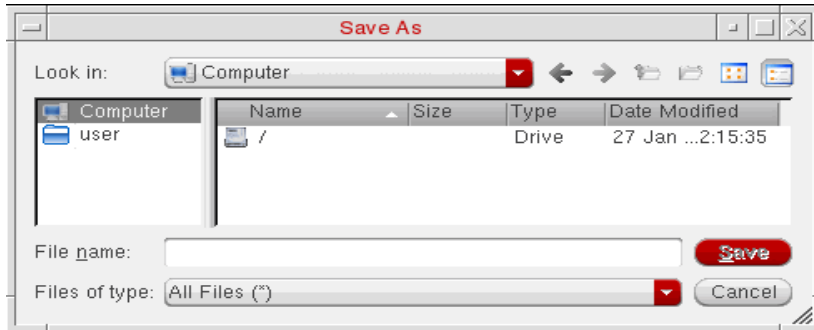
[Searching a File for Specific Text](#)

Saving a Text File

To save a text file:

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.

Related Topics

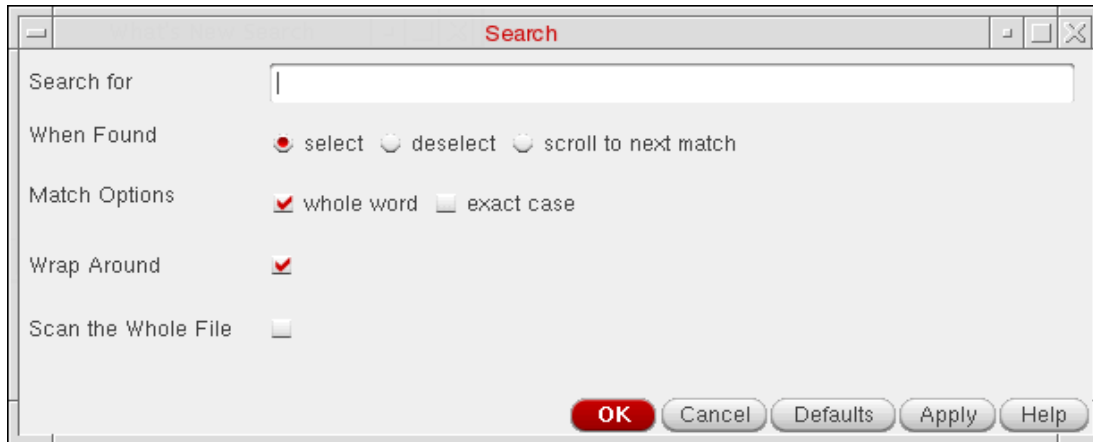
[Save As Form](#)

Searching a File for Specific Text

To search a file for a specific text string:

1. Choose *Edit – Find*.

The Search form appears.



2. In the *Search For* field, type the search string.
3. (Optional) Select a desired option from *When Found*.
4. (Optional) Select one or more of the *Match Options*.
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.

Related Topics

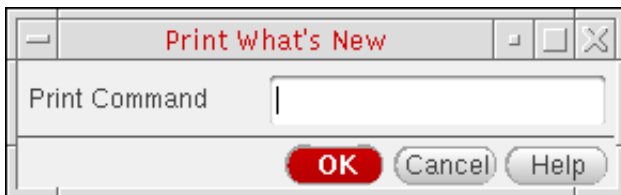
[Search Form](#)

Printing and Refreshing a File

To print the contents of a file in a text window:

1. Choose *File – Print*.

The Print What's New 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.

Depending on the application from where the Text Window is opened, the *Print* option can be unavailable.

To refresh the contents of the current file:

- ➔ Choose *File – Refresh*.
The contents of file are refreshed.

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.

Related Topics

[Saving a Text File](#)

Virtuoso Studio Design Environment User Guide

Text Window Options

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 has limited the total number of string-based objects (`oaShapes`) to this theoretical limit of 4,294,967,296 and the total number of named objects to around 2,147,483,648.

The table below lists the most common design data types and the maximum number of each type that can be safely created.

Object Type	Limit	Notes
<code>oaAppObject</code>	4,294,967,295	
<code>oaBlockage</code>	4,294,967,295	Area blockages with 4-pt polygons
<code>oaInst</code>	2,147,483,647	Auto-named and named scalarInsts
<code>oaInstTerm</code>	2,147,483,647	
<code>oaNet</code>	2,147,483,647	Auto-named and named scalarInsts
<code>oaPin</code>	2,147,483,645	
<code>oaProp</code>	4,294,967,295	
<code>oaRoute</code>	4,294,967,295	Empty routes, no beginConn or endConn
<code>oaShape</code>	4,294,967,294	
<code>oaTerm</code>	2,147,483,647	
<code>oaVia</code>	1,431,655,765	

Related Topics

[Importing and Exporting Designs](#)

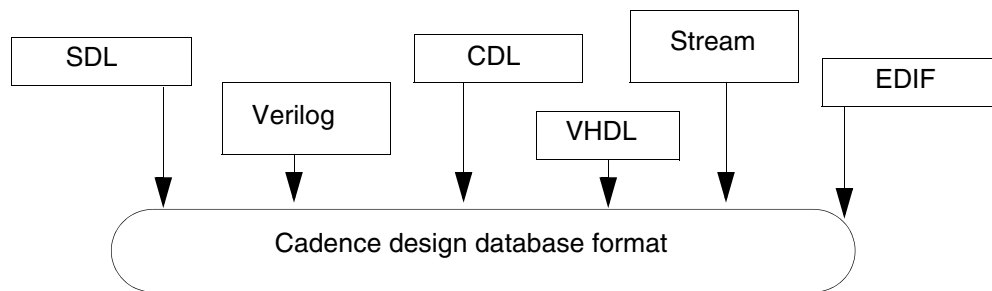
[Save As Form](#)

Importing and Exporting Designs

The Virtuoso Studio 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):

Import formats



Export formats



To import a design in another format into the Cadence design database 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.

To export a design from the Cadence design database 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.

Design Data Scaling

You can use the XScale utility to change the scale of your design data. XScale scales objects in Virtuoso Studio 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 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 and Design Exchange Format designs in SoCE typically use a DBUperUU of 2000.
- Virtuoso Studio 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 can scale all of your design and reference libraries including the PDK library.

To correct the mismatch in database units:

1. Change the DBUperUU in the technology file.
2. Use `XScale` to scale the physical elements to compensate for the change in DBUperUU.

Related Topics

[Library Exchange Format](#)

[Design Exchange Format](#)

[Technology file](#)

[Changing the DBUperUU in the Technology File](#)

[XScale Command](#)

Changing the DBUperUU in the Technology File



Do not use the `techSetDBUPerUU` function to set DBUPerUU because the technology file rules and devices are not updated properly.

To change the DBUperUU in the technology file:

1. Write the PDK technology file to an ASCII file.
2. Change the DBUperUU setting to the desired value (2000 DBUperUU for this example).
3. Load the modified ASCII technology file back into the PDK library.
4. Click *Save*.

Related Topics

[XScale Command](#)

[DBUperUU setting](#)

XScale Command

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:

Arguments	Description
<code>-mag <i>magFactor</i></code>	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.
<code>-lib <i>libName</i></code>	Name of the design library that is to be processed. You can specify only one library at a time.
<code>-cell <i>cellName</i></code>	(Optional) Name of the cellview that is to be processed. This is an optional argument. If no cell name is specified, <code>XScale</code> processes all the cells in the library.
<code>-view <i>viewName</i></code>	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. Use either <code>-view</code> or <code>-viewType</code> : You cannot use both options together.
<code>-viewType <i>viewTypeName</i></code>	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.
<code>-loadFile <i>skillFileName</i></code>	SKILL file to be loaded.

Virtuoso Studio Design Environment User Guide

Design Data Scaling

Arguments	Description
<code>-doNotScaleCellFile</code> <i>fileName</i>	Control file to be loaded.

Points to remember:

- `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.

Related Topics

[Changing the DBUperUU in the Technology File](#)

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.

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`.

Application	File
Virtuoso layout suite	<code>leBindKeys.il</code>
Place-and-route	<code>prBindKeys.il</code>
Schematic entry	<code>schBindKeys.il</code>
Layout and schematic entry	<code>leSchBindkeys.il</code>

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.

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Related Topics

[Access Keys](#)

[Menu Access Key](#)

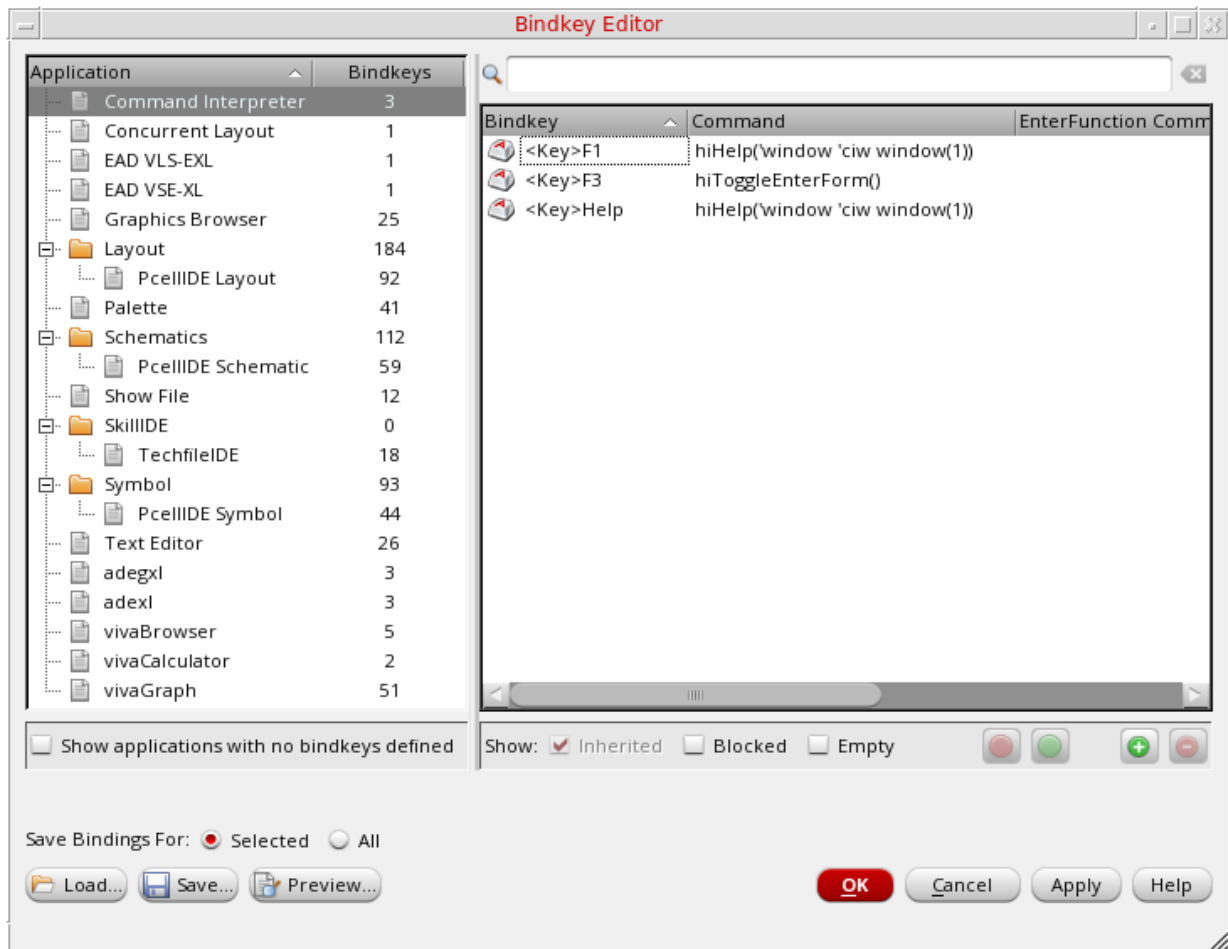
[User Preference Options](#)

Viewing the Current Bindkeys for an Application

To view current bindkey settings for an application:

1. From the Command Interpreter Window, choose *Options – Bindkeys*.

The Bindkey Editor form is displayed.



The *Application Tree*, by default, initially displays 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, lists 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:

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

- ❑ 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.

The screenshot shows a window titled "Schematics Application Bindkeys" with a menu bar (File, Edit, Help) and the Cadence logo. The main content is a text area displaying a list of bindkeys for 'Schematics'. The text is as follows:

```
;; Bindkeys for 'Schematics'
;; Inherited by:
;; * PcellIDE Schematic
;; * Schematics XL
;; * adegxl-schematic
;; * adexl-schematic
;;-----
;;
hiSetBindKeys( "Schematics" list(
  list("<DrawThru1>" "schDirectEdit(1)" "geSingleSelectBox()")
  list("<DrawThru2>" "hiDynamicPanGrabbing()" "hiDynamicPanGrabbing()")
  list("<DrawThru3>" "hiZoomIn()" "hiZoomIn()")
  list("<Key>#" "hiToggleAnchorMagnifier()")
  list("<Key>'" "hiUpdateMagOptions()")
  list("<Key>." "hiToggleMagnifier()")
  list("<Key>5" "schHiRouteFlightLine()")
  list("<Key>8" "schHiHiliteLabel(\"instance\" \"on\")")
  list("<Key>9" "geEnterAddNetProbe()")
  list("<Key>=" "schHiOpenSymbolOrSchematicView(geGetEditCellView() hi)
  list("<Key>A" "geSingleSelectPoint()")
  list("<Key>B" "schHiCreateBlockInst()")
  list("<Key>BackSpace" "schHiDelete()" "deletePoint()")
  list("<Key>C" "schHiCopy()")
)
```

Related Topics

[Configuring Application Bindkeys](#)

[Bindkey Editor Form](#)

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.
- 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 Studio 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 can 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	Usage
<i>F1</i>	Used for Help
<i>F11</i>	Reserved
<i>F12</i>	Reserved

Related Topics

[hiBindKeyModifiers](#)

Configuring Application Bindkeys

You can override bindkeys inherited from the root application by defining a new bindkey for an inherited application.

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.

Adding a Bindkey

1. In the *Application Tree*, to the left of the Bindkey Editor form, select the application for which you want to create a new bindkey.
2. Select the *Add binding* button to add the new bindkey.

You need to press the new bindkey combination.

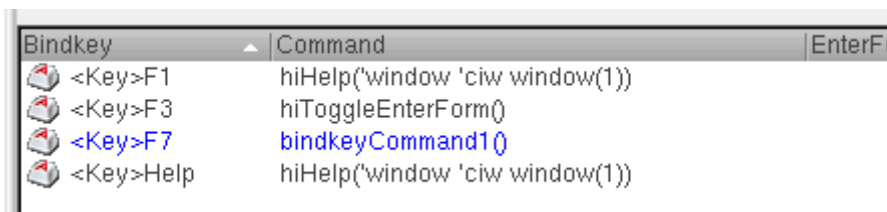
If you have more than one application selected in the *Application Tree*, an additional *Application* column gets added to the table, specifying what application a bindkey belongs to.

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 gets 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*.

The new bindkey gets displayed in blue text in the *Bindkey Table* indicating that it has been newly created, but not yet saved.



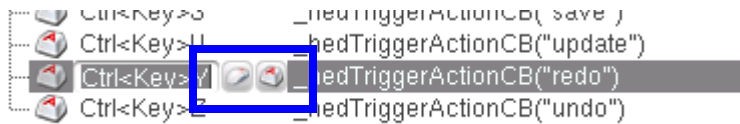
Bindkey	Command	EnterF
<Key>F1	hiHelp('window 'ciw window(1))	
<Key>F3	hiToggleEnterForm()	
<Key>F7	bindkeyCommand1()	
<Key>Help	hiHelp('window 'ciw window(1))	

5. Click the *Save* button.

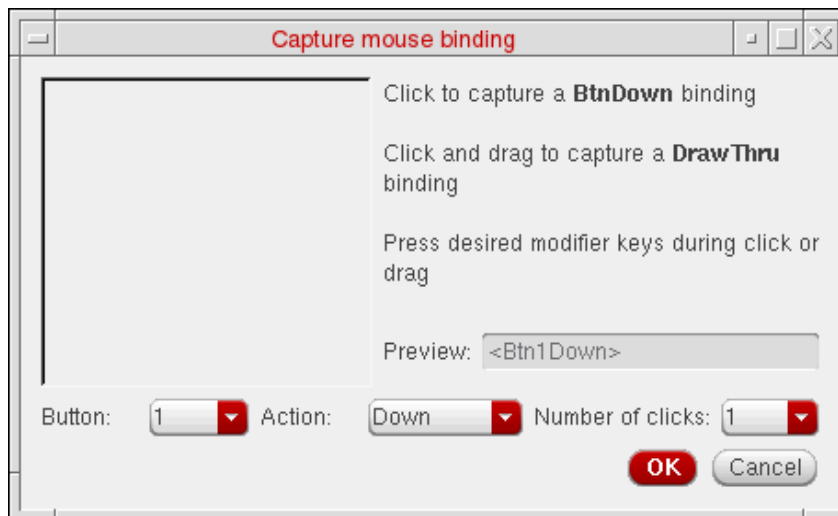
The Save Bindkeys form is displayed where you can now save the application bindkeys to file for reuse.

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.



Selecting the *Mouse* icon displays the Capture mouse binding form while selecting the *Key* icon displays the Capture key binding form.



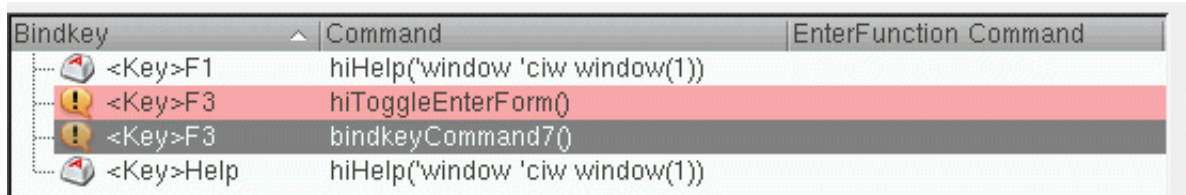
Note: Perform the mouse binding to be captured within the blank window frame area in the left of the form.

Duplicate Bindkeys

To create a new bindkey you need to enter a bindkey combination that already exists. A *Duplicate bindkeys detected* message box gets displayed.

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* then shows that there is a duplicate bindkey issue.



Bindkey	Command	EnterFunction Command
<Key>F1	hiHelp('window 'ciw window(1))	
<Key>F3	hiToggleEnterForm()	
<Key>F3	bindkeyCommand7()	
<Key>Help	hiHelp('window 'ciw window(1))	

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 gets 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.

When a bindkey is deleted from the *Bindkey Editor*, rather than it being removed completely, an empty bindkey is left in its place. 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 stores 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.

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

2. In the *Application Tree*, select the application(s) that you want to edit bindkeys in.

The *BindKey Table* gets updated 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*) allows you to edit the bindkey directly in the *BindKey Table*.

BindKey	Command	EnterFunction Command	Application
<Key>F1	hiHelp('window 'ciw window(1))		Command Interpreter
<Key>F3	hiToggleEnterForm()		Command Interpreter
<Key>Help	hiHelp('window 'ciw window(1))		Command Interpreter

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.

Related Topics

[hiConfigureBindKeys](#)

[Bindkey Editor Form](#)

Format for Key and Mouse Bindings

The following format is used for key and mouse binding modifiers:

`[modifiers]<operator>detail[(occurrences)] [EF]`

where:

Binding	Description
<i>modifier</i>	<p>(Optional) Specifies the modification key that get used with the bound key(s) or mouse button.</p> <p>Valid values are:</p> <p>Alt</p> <p>Shift</p> <p>Ctrl</p> <p>Meta</p> <p>None</p> <p>Super (or Hyper): 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 <code>xmodmap</code>, where, for example, you can re-define the use of the ride-sided Alt key (<code>Alt_R</code>) for use as a Super or Hyper modifier and assign that to an unused modifier bit.</p> <p>After the Super or Hyper modifiers are defined, they also get listed by <code>hiBindKeyModifiers</code>.</p>
<i>operator</i>	<p>Specifies the bound key or mouse button</p> <p>Surrounding angle brackets (<>) are required.</p>
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

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Binding	Description
<code>Btn9Down</code>	Thumb button farthest from the wrist
<code>DrawThru1</code>	Click and hold left mouse button while moving mouse
<code>DrawThru2</code>	Click and hold middle mouse button while moving mouse
<code>DrawThru3</code>	Click and hold right mouse button while moving mouse
<code>DrawThru8</code>	Click and hold closest thumb button while moving mouse
<code>DrawThru9</code>	Click and hold farthest thumb button while moving mouse
<code>detail</code>	Specifies the name of the keyboard key, such as <code>A</code> , <code>B</code> , <code>F3</code> , <code>Tab</code> , or <code>Del</code>
<code>occurrences</code>	(Optional) Specifies the number of times to use a mouse button. Should not be specified for <code>Btn4Down</code> or <code>Btn5Down</code> . 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.
<code>EnterFunctionCommand</code>	(Optional) Indicates that the binding is effective during a user entry function (such as drawing a rectangle).

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 `Z` and `Shift+Z` to pan and zoom during that operation.

This is achieved where:

- The Arrow keys pan in the appropriate direction by calling `hiAbsolutePan`
- `Z` calls `hiZoomInAtMouse`
- `Shift+Z` calls `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.

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Related Topics

[hiAbsolutePan](#)

[hiZoomInAtMouse](#)

[hiZoomOutAtMouse](#)

[hiZoomAbsoluteScale](#)

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.

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, Oblique, Oslash (same as Oblique), 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, oblique (same as oslash), ugrave, uacute, ucircumflex, udiaeresis, yacute, thorn, ydiaeresis

These names are case-sensitive.

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Additionally, the names of regular ASCII symbols can be used in place of the symbol (for space, the name must be used):

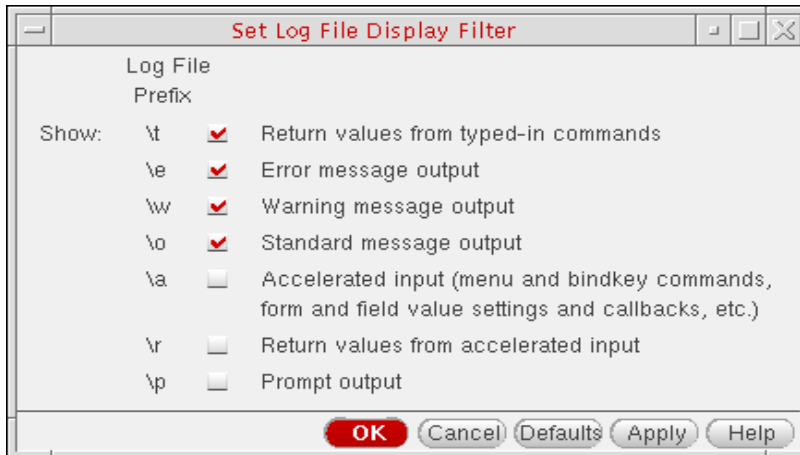
```
' ' = "space"
! = "exclam"
" = "quotedbl"
# = "numbersign"
$ = "dollar"
% = "percent"
& = "ampersand"
' = "apostrophe"
' = "quoteright" // deprecated
( = "parenleft"
) = "parenright"
* = "asterisk"
+ = "plus"
, = "comma"
- = "minus"
. = "period"
/ = "slash"
: = "colon"
; = "semicolon"
< = "less"
= = "equal"
> = "greater"
? = "question"
@ = "at"
[ = "bracketleft"
\ = "backslash"
] = "bracketright"
^ = "asciicircum" // a/k/a "circumflex"
_ = "underscore"
` = "grave"
` = "quoteleft" // deprecated
{ = "braceleft"
| = "bar"
} = "braceright"
~ = "asciitilde"
```

Determining SKILL Function for a Menu Command

To determine the SKILL function associated with a menu command:

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*.
4. Click the menu command whose SKILL function you want to know.

SKILL functions appear in the CIW output area.

The SKILL function associated with the menu command appears in the CIW. This function is the last one to appear in your log file. The default log file name is `CDS.log` in your home directory (unless you specified otherwise).

Related Topics

Set Log File Display Filter Form

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

Arguments	Description
<i>t_application</i>	Specifies the name of the application or window for which you want to set a bindkey
<i>t_key</i>	Specifies the name of a key, key sequence, or mouse button
<i>t_SKILL_cmd</i>	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()")
```

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.

Related Topics

[Restrictions to Setting Bindkeys](#)

Control 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: Return, Escape, Delete, BackSpace, Space, Tab, Up, keypad keys, 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 the following function which has a space between the first parenthesis and the `getCurrentWindow` function:

```
hiGetBannerMenus( getCurrentWindow() )
```

and you bind a key to:

```
hiGetBannerMenus(getCurrentWindow())
```

The binding works, but the bindkey assignment does not appear.

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.

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Here are some common assignments:

Bindkey Assignment	Keys Used	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

Related Topics

[Rules for Bindkey Use with Keypads](#)

Default Keybindings for Text Fields

Virtuoso 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.

Related Topics

[Default Keybindings for Forms](#)

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	<p>If the the text is selected and the <code>Esc</code> key is pressed, it clears the selection. However, if no text is selected, the <code>Esc</code> key clears the text from the input area.</p> <p>In addition, the <code>Esc</code> key closes the regular forms and hides the option forms, which are displayed by pressing the <code>F3</code> key, when the focus is on them. In case of option form, the <code>Esc</code> key only hides the form when pressed once and does not cancel the command. However, if the <code>Esc</code> key is pressed twice, the first press hides the form and the second press cancels the command.</p>
Ctrl+Esc Ctrl+C	A parser error gets generated.
Ctrl+Insert Ctrl+C	<p>Copies selected text to clipboard when text is selected in the input or output areas.</p> <p>Use <code>F16</code>, the copy key, on Sun keyboards.</p>
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

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Key Sequence	Action
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 <i>History In Place</i> 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	<ul style="list-style-type: none"> ■ If syntax highlighting is off, executes command or sends partial command to SKILL. ■ If syntax highlighting is on, Ctrl+J and Ctrl+M it inserts a new line. ■ If syntax highlighting is on and the <i>Enter Key Executes Command</i> option is selected, Enter and Return executes the command and Shift+Enter and Shift+Return inserts a new line, else if the <i>Enter Key Executes Command</i> option is not selected, Enter and Return inserts a new line and Shift+Enter and Shift+Return executes the command. <p>If syntax highlighting is on, the command is sent to the SKILL parser if the syntax is complete, otherwise, the CIW syntax checker produces a beep sound and do nothing. If syntax highlighting is off, the command or partial command is sent to the SKILL parser and the parser executes the command once it receives a complete command.</p>
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. This is only available if syntax highlighting on
Shift+Alt+J Ctrl+Backspace	<p>Joins previous command block (only available if syntax highlighting on)</p> <p>Use Ctrl+Backspace if the cursor is at the start of the command block</p>
Down Arrow Ctrl+N	Moves the cursor down one line
Left Arrow Ctrl+B	Moves left one character

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Key Sequence	Action
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 of the 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

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Key Sequence	Action
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 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

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Key Sequence	Action
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)

Related Topics

[Default Keybindings for Text Fields](#)

Default Keybindings for Forms

The following default bindings are used for forms:

Key Sequence	Action
Alt + O	Click the <i>OK</i> button
Alt + A	Click the <i>Apply</i> button
Alt + D	Click the <i>Defaults</i> button
Alt + H	Click the <i>Help</i> or <i>Hide</i> button
Alt + C	Click the <i>Cancel</i> or <i>Close</i> button
Alt + M	Click the <i>More</i> button
Alt + Y	Click the <i>Yes</i> button
Alt + N	Click the <i>No</i> button
Alt + L	Click the <i>Last</i> button (Available in option forms)
Alt + Q	Click the <i>Quit</i> button (Available in option forms)

Related Topics

Default Keybindings for Text Fields

Rules for Bindkey Use with Keypads

When using bindkeys with keypads, the following rules should be considered:

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) 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 fails to invoke anything. Rather, it would only invoke a bindkey for KP_7.

2. If NumLock is *off*, then non-numeric keypad binding is searched 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 invokes the bindkey for Home. If Home and KP_7 are both bound, but not KP_Home, then the bindkey for KP_7 gets invoked. And, finally, if KP_Home is bound, KP_Home gets invoked.

Additionally, if NumLock is *off*, then the keys that surround the keypad (for example, KP_Divide, KP_Multiply, and KP_Enter) works the same as if NumLock were *on*.

The symbol keys do not match the printable characters ("/", "*", and so on), however the Enter key on the keypad matches "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 matches "KP_Delete" if that is defined, then "KP_Decimal", then finally "Delete". If NumLock is *on*, it only matches "KP_Decimal".

When working on remote servers, work with the CapsLock key off to avoid impacting the behavior of bindkeys.

Related Topics

[Control Bindkey Display on Menus](#)

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 displays the contents of the *File* banner menu.

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.

When you have displayed a menu’s content further menu access keys may be used to run any of the sub-menu options listed.

When content of a menu 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` to display the content of the *Launch* menu.
2. Select the access key `M` to display the contents of the *Mixed Signal Options* sub-menu, where the *AMS* option can be found.
3. Select the access key `A` to add the *AMS* menu to the banner menu.
4. Select the access keys `Alt + A` to display the content of the *AMS* menu on the banner menu.

Related Topics

[Quick Reference - Menu Access Keys](#)

Virtuoso Studio Design Environment User Guide

Bindkeys and Access Keys

Design Environment Variables

This topic provides information on the names, descriptions, and graphical user interface equivalents for the design environment variables.

CDBA Environment Variables

Performance Environment Variables

User Interface Environment Variables

CDBA Environment Variables

This section provides information on the names, descriptions, and graphical user interface equivalents for the CDBA environment variables.

- AlternateFoundryCG
- dbAddCellNameToInstNamePrefix
- dbAllowStdViaCutLayerOverride
- dbArrayInstNamePrefix
- dbEnableRouteObservers
- dbFigGroupNamePrefix
- dbInstNamePrefix
- dbLogPcellWarnings
- dbNumCPU
- dbUndoAcrossPurge
- dbUndoAcrossSave
- dbUpdateCellNameInInstNamePrefixDuringRemaster
- defaultAttachTech
- disablePartialRead
- noDetailedRowCol
- [noTechUpRev](#)
- propsToAppend
- resetLibList
- resetOnRemaster

AlternateFoundryCG

```
cdba.layout AlternateFoundryCG string t_alternateFoundryConstraintGroupName
```

Description

Specifies the name of the constraint group that contains the foundry constraints. All constraint groups that are intended to be used as foundry constraint groups must be defined using the alternateFoundry constraint group definition. In the Layout Editor Options form, the Editor field displays only those constraint groups that are defined with the alternateFoundry constraint group definition.

The default value is " ", in which case, the foundry constraint group is used.

GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Foundry Constraints</i>

Examples

```
envGetVal("cdba.layout" "AlternateFoundryCG")  
envSetVal("cdba.layout" "AlternateFoundryCG" 'string "my_foundry")
```

Related Topics

[Constraint Group Definitions](#)

[Layout Editor Options Form](#)

copyMPAttributes

```
cdba copyMPAttributes boolean { t | nil }
```

Description

Specifies whether the coloring information from source objects is copied as-is to the corresponding destination objects when copying or using MakeCell.

The default is `nil`.

GUI Equivalent

Command	<i>Options – Multiple Patterning</i>
Form Field	<i>Maintain color while copying</i>

Examples

```
envGetVal("cdba" "copyMPAttributes")  
envSetVal("cdba" "copyMPAttributes" 'boolean t)
```

Related Topics

[Multiple Patterning Options](#)

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, where, *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)
```

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 nil)
```

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")
```

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.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbEnableRouteObservers")  
envSetVal("cdba" "dbEnableRouteObservers" 'boolean t)
```

Related Topics

[dbGetRouteAuthor](#)

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")
```

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")
```

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" "dbLogPcellWarnings")  
envSetVal("cdba" "dbLogPcellWarnings" 'boolean t)
```

Related Topics

[Copying the Current Cellview to a New Session Window](#)

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)
```

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.undo" "dbUndoAcrossPurge")  
envSetVal("cdba.undo" "dbUndoAcrossPurge" 'boolean t)
```

Related Topics

[Closing Cellviews in Virtuoso](#)

[Close and Purge Data Form](#)

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.undo" "dbUndoAcrossSave")  
envSetVal("cdba.undo" "dbUndoAcrossSave" 'boolean t)
```

Related Topics

[Save Cellviews Form](#)

dbUpdateCellNameInInstNamePrefixDuringRemaster

```
cdba dbUpdateCellNameInInstNamePrefixDuringRemaster boolean { t | nil }
```

Description

Adds cell name as the prefix of the name of the instance being remastered. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "dbUpdateCellNameInInstNamePrefixDuringRemaster")  
envSetVal("cdba" "dbUpdateCellNameInInstNamePrefixDuringRemaster" 'boolean t)
```

Related Topics

[resetOnRemaster](#)

[resetLibList](#)

defaultAttachTech

`cdba defaultAttachTech string t_techLibraryName`

Description

Specifies the technology library to be selected by default when the user opts to attach a new library to an existing technology library.

GUI Equivalent

Command	<i>File – New – Library</i> (In CIW and Library Manager)
Form Field	<i>Attach to an existing technology library</i>

Examples

```
envGetVal("cdba" "defaultAttachTech")
envSetVal("cdba" "defaultAttachTech" 'string "Lib1")
```

Related Topics

[Attaching a New Library to an Existing Technology Library](#)

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 DFI 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")
```

Related Topics

[Design Object Limits in OpenAccess](#)

maxMasterSize

`cdba.hierCache maxMasterSize float maximum_size`

Description

Specifies the maximum size, in megabytes, for a regular master that can be saved to the cache. If a regular master has a larger size than the specified value, then it cannot be included in the cache.

The default value is 0, which means there is no maximum size. All regular masters can be saved to the cache.

GUI Equivalent

None

Examples

```
envGetVal("cdba.hierCache" "maxMasterSize")  
envSetVal("cdba.hierCache" "maxMasterSize" 'float 3)
```

Related Topics

[sessionUsageType](#)

[verbosity](#)

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))
```

Virtuoso Studio Design Environment User Guide

Design Environment Variables

```
(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)
)
```

Virtuoso Studio Design Environment User Guide

Design Environment Variables

```
; 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

[dbGetTrueOverlaps](#)

noTechUpRev

```
cdba noTechUpRev boolean { t | nil }
```

Description

Disables the upgrade process that normally takes place when a technology library is opened for the first time in a newer version of Virtuoso. This can be used to prevent the insertion of Virtuoso system-reserved LPPs into technology libraries created outside Virtuoso.

By default, this is set to `nil` and the upgrade process happens as needed.

GUI Equivalent

None

Examples

```
envGetVal("cdba" "noTechUpRev")  
envSetVal("cdba" "noTechUpRev" 'boolean t)
```

Related Topics

[CDBA Environment Variables](#)

propsToAppend

```
cdba.copyUpdater propsToAppend string "propertyName"
```

Description

Prepends the value of the source library property to the value of the destination library property when both the source and destination libraries have the same property name, and this property name is mentioned in the environment variable. This feature is functional only when the *g_overWrite* argument of the *ccpCopy* SKILL function is set to *nil*.

The default is " " .

You must specify this environment variable in the local *.cdsenv* file. This feature does not work if *propsToAppend* is set using the *envSetVal* SKILL function.

If values of more than one property are to be appended during a copy operation, specify the names of these properties in the environment variable by adding a space between each property name, as shown below:

```
cdba.copyUpdater propsToAppend string "propName1 propName2"
```

GUI Equivalent

None

Example

This example shows how to prepend the value of the source library property to the value of the destination library property using *propsToAppend*:

```
; Add prop to the source library.
propBag = dbOpenBag(srcLib "w")
prop = dbCreateProp(srcLib "prop" "string" "srcValue")
dbSaveBag(propBag)
dbPurgeBag(propBag)

; Add same named prop to the destination library.
propBag = dbOpenBag(dstLib "w")
prop = dbCreateProp(dstLib "prop" "string" "dstValue")
dbSaveBag(propBag)
dbPurgeBag(propBag)
```


Virtuoso Studio Design Environment User Guide

Design Environment Variables

```
list1 = gdmCreateSpecList()
list2 = gdmCreateSpecList()
spec1 = gdmCreateSpec("srcLib" "" "" "" "CDBA")
spec2 = gdmCreateSpec("dstLib" "" "" "" "CDBA")
gdmAddSpecToSpecList(spec1 list1)
gdmAddSpecToSpecList(spec2 list2)

ccpCopy(list1 list2 nil 'CCP_EXPAND_ALL)

; Query the prop value on the destination library.
\p >
\i dstLib~>prop~>value
\t ("srcValue dstValue" "cdsDefTechLib")
\p >
```

Related Topics

[ccpCopy](#)

sessionUsageType

```
cdba.hierCache sessionUsageType cyclic {"none" | "saveOnly" | "loadOnly" |  
    "saveAndLoad"}
```

Description

Specifies the hierarchical cache operations that are allowed in the current session. Both the session usage and design usage must permit an action before it can be performed.

The default value is `saveAndLoad`.

- `none`: No operations are specified.
- `saveOnly`: Indicates that saving a hierarchical cache is permitted in the current session. A hierarchical cache is always saved with the cellview, which is mostly the top design in the hierarchy.
- `loadOnly`: Indicates that loading the design hierarchy from a hierarchical cache is permitted in the current session.
- `saveAndLoad`: Indicates that both save and load operations using the hierarchical cache are permitted.

GUI Equivalent

None

Examples

```
envGetVal("cdba.hierCache" "sessionUsageType")  
envSetVal("cdba.hierCache" "sessionUsageType" 'cyclic "saveOnly")  
envSetVal("cdba.hierCache" "sessionUsageType" 'cyclic "loadOnly")
```

Related Topics

[maxMasterSize](#)

[verbosity](#)

resetLibList

```
cdba.params resetLibList string "libraryName"
```

Description

Specifies the library on which the remastering has to be done.

The default is "".

GUI Equivalent

None.

Examples

```
envGetVal("cdba.params" "resetLibList")  
envSetVal("cdba.params" "resetLibList" 'string "myLib")
```

Related Topics

[dbUpdateCellNameInInstNamePrefixDuringRemaster](#)

[resetOnRemaster](#)

resetOnRemaster

```
cdba.params resetOnRemaster boolean { t | nil }
```

Description

Resets the new master CDF parameter value to the default value.

When `resetOnRemaster` is set, editing the master for multiple instances is disabled. This is because remastering requires the changes to be applied immediately. The tool only enables editing when the value of the *Apply To* field is set to "only current" and the current cellview is writable.

This environment variable controls the behavior of device remastering through `dbRemasterAnyInst()`, which can be invoked by modifying the library or cell lists in the Edit Object Properties form.

Possible values are:

- `t`: the remastering flow is performed to allow the database properties to be updated.
- `nil`: the instance is remastered but no database properties are modified. This is the default.

GUI Equivalent

None

Examples

```
envGetVal("cdba.params" "resetOnRemaster")  
envSetVal("cdba.params" "resetOnRemaster" 'boolean t)
```

Related Topics

[dbUpdateCellNameInInstNamePrefixDuringRemaster](#)

[resetLibList](#)

usePerDesignAFCG

```
cdba.layout usePerDesignAFCG boolean { t | nil }
```

Description

Specifies whether the name of the constraint group that contains the foundry constraints should be specified for each design. All constraint groups that are intended to be used as foundry constraint groups must be defined using the alternateFoundry constraint group definition.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("cdba.layout" "usePerDesignAFCG")  
envSetVal("cdba.layout" "usePerDesignAFCG" 'boolean t)
```

Related Topics

[AlternateFoundryCG](#)

[Constraint Group Definitions](#)

[Layout Editor Options Form](#)

verbosity

```
cdba.hierCache verbosity cyclic {"none" | "low" | "medium" | "high"}
```

Description

Specifies the detail level of the output messages displayed in the CIW during a hierarchical cache operation. The default value is `none`.

- `none`: No informational messages regarding the cache are displayed. Only exceptions and warning messages are displayed.
- `low`: Displays the informational messages that indicate if a cache is saved, updated, or used to load the design hierarchy.
- `medium`: Displays the informational messages that contain a summary of the cache sections, which are based on the design type.
- `high`: Displays the informational messages that contain details on individual designs during a cache operation. It includes the messages that lists the designs that are excluded from the cache or already opened when loading the cache.

Each verbosity level includes the messages from the levels below it. For example, if the `high` verbosity level is enabled, then messages corresponding to `high`, `medium`, and `low` verbosity levels are displayed during the hierarchical cache operation.

GUI Equivalent

None

Examples

```
envGetVal("cdba.hierCache" "verbosity")  
envSetVal("cdba.hierCache" "verbosity" 'cyclic "high")
```

Related Topics

[sessionUsageType](#)

[maxMasterSize](#)

closeDataOption

```
ddserv.ciw closeDataOption cyclic { "All" | "None" }
```

Description

Selects all the objects or data listed in the Close and Purge Data form during or after the initialization. The default value is `None`.

- `All`: Selects all the data.
- `None`: No data is selected.

GUI Equivalent

Command: *CIW – File – Close Data*

Field: *Select one or more objects to close and purge from virtual memory*

Examples

```
envGetVal("ddserv.ciw" "closeDataOption")  
envSetVal("ddserv.ciw" "closeDataOption" 'cyclic "All")
```

Related Topics

[closeDataSaveOption](#)

[Close and Purge Data Form](#)

closeDataSaveOption

```
ddserv.ciw closeDataSaveOption cyclic { "prompt" | "discard" | "save" }
```

Description

Specifies the behavior for the close and purge data functionality to manage modified data for the files selected to close using `closeDataOption`. The default value is `prompt`.

- `prompt`: Prompts to save data before closing the window.
- `discard`: Discards the changes and closes the window.
- `save`: Saves the changes and closes the window.

GUI Equivalent

Command: *CIW – File – Close Data*

Field: *For all modified data*

Examples

```
envGetVal("ddserv.ciw" "closeDataSaveOption")  
envSetVal("ddserv.ciw" "closeDataSaveOption" 'cyclic "discard")  
envSetVal("ddserv.ciw" "closeDataSaveOption" 'cyclic "save")
```

Related Topics

[closeDataOption](#)

[Close and Purge Data Form](#)

importDuplicateBookmarks

```
ddserv importDuplicateBookmarks boolean { t | nil }
```

Description

Imports all the duplicate bookmarks which means any bookmarks that share the same name and callbacks between the imported bookmark file and the current bookmark file gets imported.

The default is `t`.

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.

GUI Equivalent

None

Examples

```
envSetVal ("ddserv" "importDuplicateBookmarks")  
envSetVal ("ddserv" "importDuplicateBookmarks" 'boolean nil)
```

Related Topics

[Managing Bookmarks](#)

numThreads

`ddserv.refresh numThreads int number_of_threads`

Description

Specifies the maximum number of threads used to refresh the current Virtuoso session. The actual number of threads used may vary depending on the task to be performed and the system load. These threads can be used to speed up the refresh operation for the library or hierarchy whenever required.

The default value is 0, which means Virtuoso is responsible for determining whether to use multithreading and the number of threads to be used in the process.

If the value is set to 1, multithreading is not used for the refresh operation.

Note: Multithreading requires a multicore CPU.

GUI Equivalent

None

Examples

```
envGetVal("ddserv.refresh" "numThreads")  
envSetVal("ddserv.refresh" "numThreads" 'int 2)
```

addVLSEXLUserTriggersToMXL

```
designEditor.appReg addVLSEXLUserTriggersToMXL boolean { t | nil }
```

Description

Calls all the user menu and post-install triggers registered on `maskLayoutEXL` automatically when installing Layout MXL.

The default value is `nil`.

GUI Equivalent

None

Example

```
envGetVal("designEditor.appReg" "addVLSEXLUserTriggersToMXL")  
envSetVal("designEditor.appReg" "addVLSEXLUserTriggersToMXL" 'boolean t)
```

Related Topics

[addVLSXLUserTriggersToMXL](#)

addVLSXLUserTriggersToMXL

```
designEditor.appReg addVLSXLUserTriggersToMXL boolean { t | nil }
```

Description

Calls all the user menu and post-install triggers registered on `maskLayoutXL` automatically when installing Layout MXL.

The default value is `nil`.

GUI Equivalent

None

Example

```
envGetVal("designEditor.appReg" "addVLSXLUserTriggersToMXL")  
envSetVal("designEditor.appReg" "addVLSXLUserTriggersToMXL" 'boolean t)
```

Related Topics

[addVLSEXLUserTriggersToMXL](#)

resetUndoOnDescending

```
designEditor.hierarchy resetUndoOnDescending boolean { t | nil }
```

Description

Resets the undo stacks on descending in the design hierarchy.

The default value is t.

GUI Equivalent

None

Examples

```
envGetVal("designEditor.hierarchy" "resetUndoOnDescending")
envSetVal("designEditor.hierarchy" "resetUndoOnDescending" 'boolean nil)
```

Related Topics

[Starting Virtuoso Studio](#)

ignoreAppTierInHistory

```
designEditor.history ignoreAppTierInHistory boolean { t | nil }
```

Description

Ignores the application tier saved in history when an item is selected from the history menu in Virtuoso Studio Design Environment. When set to `nil`, the application tier is considered.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("designEditor.history" "ignoreAppTierInHistory")
envSetVal("designEditor.history" "ignoreAppTierInHistory" 'boolean nil)
```

Related Topics

[Cellview History](#)

cellviewModifiedIndicator

```
designEditor.window cellviewModifiedIndicator string "NewIndicatorCharacter"
```

Description

Customizes the modified cellview indicator in the title. The default is *, which means no value is set.

GUI Equivalent

None

Examples

```
envGetVal("designEditor.window" "cellviewModifiedIndicator")  
envSetVal("designEditor.window" "cellviewModifiedIndicator" 'string "%c")
```

Related Topics

[iconNameFormat](#)

[windowNameFormat](#)

fitViewAttempts

`designEditor.window fitViewAttempts int number_of_attempts`

Description

Specifies 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 to 0. You can set this number based on your working environment.

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

Default value: 1

GUI Equivalent

None

Examples

```
envGetVal("designEditor.window" "fitViewAttempts")
envSetVal("designEditor.window" "fitViewAttempts" 'int 2)
```

Related Topics

[Opening a Cellview from the CIW](#)

iconNameFormat

```
designEditor.window iconNameFormat string "Format"
```

Description

Customizes an icon title by specifying a format. The default is "%C".

GUI Equivalent

None

Examples

```
envGetVal("designEditor.window" "iconNameFormat")  
envSetVal("designEditor.window" "iconNameFormat" 'string "%v")
```

Related Topics

[cellviewModifiedIndicator](#)

[windowNameFormat](#)

Virtuoso Studio Design Environment User Guide

Design Environment Variables

windowNameFormat

`designEditor.window windowNameFormat string "winTitle"`

Description

Customizes the window title. The default is "%a %m: %l %c %v %x".

Valid values are:

Value	Description
%a	<p>Specifies the application name in a window title.</p> <p>This means that if the string returned by <code>envGetVal("designEditor.window" "windowNameFormat" 'string)</code> does not contain sub-string "%a", it is automatically added in the front of the return value.</p> <p>For example, if a user sets <code>envSetVal("designEditor.window" "windowNameFormat" 'string "%l %c %v")</code> a layout session window title is updated to:</p> <p>Virtuoso (R) Layout Suite L <lib-name> <cell-name> <view-name></p> <p>But if it is set to "%l %c %v, %a", the window title reads as follows:</p> <p><lib-name> <cell-name> <view-name>, Virtuoso (R) Layout Suite L</p>
%m	Specifies the access mode, such as editing.
%l	Specifies the library name.
%c	Specifies the cell name.
%v	Specifies the view name.
%x	Specifies the configuration, such as, <lib> <cell> <view>.
%L	Specifies the top library name.
%C	Specifies the top cell name.
%V	Specifies the top view name.

Virtuoso Studio Design Environment User Guide

Design Environment Variables

Value	Description
%h	Specifies the hierarchy path, which is the value of the <code>geGetInstHier</code> SKILL function.

GUI Equivalent

None

Examples

```
envGetVal("designEditor.window" "windowNameFormat")  
envSetVal("designEditor.window" "windowNameFormat" 'string "%l %c %v, %a")
```

Related Topics

[geGetInstHier](#)

[iconNameFormat](#)

[cellviewModifiedIndicator](#)

Performance Environment Variables

This section provides information on the names, descriptions, and graphical user interface equivalents for the Health Monitor environment variables.

- autoLogLatency
- cpuUtilization
- deleteHMLog
- enableExpertMode
- fileTag
- gdbPath
- installAtStartup
- installPath
- launchUI
- logDir
- memUtilization
- openReportForm
- pinnedOffsetX
- popUpLog
- postReportScript
- preReportScript
- pstackPath
- stracePath
- sysMonitorPath
- sysPulseRefreshInterval
- sysPulseYellowLightAlert
- toolbarAutohide
- toolbarBorderless
- toolbarNativeWM

Virtuoso Studio Design Environment User Guide

Design Environment Variables

- virtuosoPulseRefreshInterval
- virtuosoPulseYellowLightAlert

autoLogLatency

```
perf.hm autoLogLatency int time_interval
```

Description

Specifies the threshold level in seconds of continuous Virtuoso Studio alerts or system event alarms to start an auto capture of logs. You can set the value to 0 to stop auto logging.

The default is 20.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *Virtuoso Pulse – Time to Start Auto Logging*

Examples

```
envGetVal("perf.hm" "autoLogLatency")  
envSetVal("perf.hm" "autoLogLatency" 'int 100)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[virtuosoPulseYellowLightAlert](#)

[sysPulseRefreshInterval](#)

cpuUtilization

```
perf.hm cpuUtilization int CPU_utilization_percentage
```

Description

Specifies the threshold level in percentage of high CPU utilization to report a concentrated pulse. You can set the value to 0 to stop the CPU alert.

The default is 70.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *System Pulse – CPU Utilization*

Examples

```
envGetVal("perf.hm" "cpuUtilization")  
envSetVal("perf.hm" "cpuUtilization" 'int 90)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[virtuosoPulseYellowLightAlert](#)

[sysPulseRefreshInterval](#)

deleteHMLog

```
perf.hm deleteHMLog cyclic {"always" | "never" | "ask"}
```

Description

Specifies whether to delete the log files created by Health Monitor after exiting the current Virtuoso Studio session.

- **always**: Deletes the Health Monitor log files. This is the default.
- **never**: Retains the log files for debugging, consuming disk space.
- **ask**: Prompts to confirm the deletion of the Health Monitor log files.

GUI Equivalent

None

Examples

```
envGetVal("perf.hm" "deleteHMLog")  
envSetVal("perf.hm" "deleteHMLog" 'cyclic "never")
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

enableExpertMode

```
perf enableExpertMode boolean { t | nil }
```

Description

Displays the advanced debugging options on the Health Monitor form. The default is `nil`.

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Health Monitor*

Field: *Expert*

Examples

```
envGetVal("perf" "enableExpertMode")  
envSetVal("perf" "enableExpertMode" 'boolean t)
```

Related Topics

[Collecting Data Using Health Monitor Tool](#)

fileTag

```
perf fileTag string "keywords"
```

Description

Specifies the file tag to add as a suffix to the callstack log file name. The default is "" .

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Health Monitor –
Program Execution Tracing*

Field: *File Tags*

Examples

```
envGetVal("perf" "fileTag")  
envSetVal("perf" "fileTag" 'string "slow_rectangle_creation")
```

The following example shows a callstack file name with specified file tag added as a suffix:

```
/.cadence/perf/  
hc1lnx14_myuser_16206_callstacks_20201223_105512_slow_rectangle_creati  
on
```

Related Topics

[Collecting Data Using Health Monitor Tool](#)

gdbPath

```
perf gdbPath string "gdbpath"
```

Description

Specifies the full path to GDB version 7.2 or later. The default is "gdb".

Note: If a valid `gdbPath` is set in the shell environments `CDS_DEBUGGER`, `DEBUGGER`, or `GDB`, the `gdbPath` is overwritten by the information specified in the shell environment.

GUI Equivalent

None

Examples

```
envGetVal("perf" "gdbPath")  
envSetVal("perf" "gdbPath" 'string "/usr/bin/gdb")
```

Related Topics

[Crash Report Customization](#)

installAtStartup

```
perf installAtStartup boolean { t | nil }
```

Description

Specifies whether Health Monitor is installed when Virtuoso is launched. The default is `t`, which means that the tool is installed at startup.

GUI Equivalent

None

Examples

```
envGetVal("perf" "installAtStartup")  
envSetVal("perf" "installAtStartup" 'boolean nil)
```

Related Topics

[Health Monitor Overview](#)

installPath

```
perf installPath string "tarkit_install_path"
```

Description

Specifies the path to the Health Monitor installation tarkit.

The default is " ", which means that the Health Monitor installation bundled with the Virtuoso installation files is used.

GUI Equivalent

None

Examples

```
envGetVal("perf" "installPath")  
envSetVal("perf" "installPath" 'string "~/perfDiagDir")
```

Related Topics

[Health Monitor Overview](#)

launchUI

```
perf launchUI cyclic { "no" | "pinned" | "expanded" }
```

Description

Specifies the display type of the Health Monitor form.

- **no**: The Health Monitor form is not displayed. This is default.
- **pinned**: Pins the Health Monitor form as a toolbar at the top of the screen.
- **expanded**: Displays the full view of the Health Monitor form.

Use the environment variable `InstallAtStartup` with `launchUI` to control the installation of the Health Monitor form while launching Virtuoso.

Note: Changing the environment variables after launching Virtuoso has no effect.

GUI Equivalent

None

Examples

```
envGetVal("perf" "launchUI")
envSetVal("perf" "launchUI" 'cyclic "pinned")
envSetVal("perf" "launchUI" 'cyclic "expanded")
```

Related Topics

[Health Monitor Overview](#)

[pinnedOffsetX](#)

logDir

```
perf logDir string "check_log_file_path"
```

Description

Specifies the path to the Health Monitor log file. The default is "", which means that the default path `./cadence/perf` is used.

GUI Equivalent

None

Examples

```
envGetVal("perf" "logDir")  
envSetVal("perf" "logDir" 'string "~/perfDiagLog")
```

Related Topics

[Health Monitor Overview](#)

memUtilization

```
perf.hm memUtilization int memory_utilized_percentage
```

Description

Specifies the threshold level in percentage of high memory utilization to report a concentrated pulse. You can set the value to 0 to stop the memory alert.

The default is 80.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *System Pulse – Memory Utilization*

Examples

```
envGetVal("perf.hm" "memUtilization")  
envSetVal("perf.hm" "memUtilization" 'int 90)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[sysPulseRefreshInterval](#)

openReportForm

```
perf openReportForm boolean { t | nil }
```

Description

Automatically opens the report slowness form after the recording stops. The default is `t`.

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Health Monitor – Expert*

Field: *Program Execution Tracing – Open Report Form*

Examples

```
envGetVal("perf" "openReportForm")  
envSetVal("perf" "openReportForm" 'boolean nil)
```

Related Topics

[Reporting Slowness](#)

[Collecting Data Using Health Monitor Tool](#)

[Health Monitor Form](#)

pinnedOffsetX

```
perf    pinnedOffsetX int offset
```

Description

Specifies the offset in pixels to shift a pinned toolbar in the horizontal direction. The default offset is 250 pixels.

This environment variable is referenced when the environment variable `launchUI` is set to `pinned`. A negative offset value means that the toolbar is shifted to the left.

Examples

```
envGetVal("perf" "pinnedOffsetX")
envSetVal("perf" "pinnedOffsetX" 'int 300) ;shift to right
envSetVal("perf" "pinnedOffsetX" 'int -250) ;shift to left
```

Related Topics

[launchUI](#)

[Health Monitor Overview](#)

popUpLog

```
perf popUpLog boolean { t | nil }
```

Description

Opens the callstack log file automatically. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("perf" "popUpLog")  
envSetVal("perf" "popUpLog" 'boolean nil)
```

Related Topics

[Collecting Data Using Health Monitor Tool](#)

postReportScript

```
perf postReportScript string "post_Report_Script_Path"
```

Description

Customizes the post report script in order to customize the slow report. It is used for post-processing activities. For example, creating a tarball, sending email, and so on.

To execute this script, use the command argument `postReportScript -log <path_to_slowReport.log> -t <slow_time> -dir <log_directory> -exe <path_to_virtuoso_binary>`.

The default is " ".

GUI Equivalent

None

Examples

```
envGetVal("perf" "postReportScript")  
envSetVal("perf" "postReportScript" 'string "~/myPostSlowReportScript.sh")
```

Related Topics

[Reporting Slowness](#)

preReportScript

```
perf preReportScript string "pre_Report_Script_Path"
```

Description

Customizes the pre report script in order to customize the slow report. It collects additional data from the commands, such as `lsof` and `strace`, to append to the `slowReport.CDS.log`.

To execute this script, use the command argument `preReportScript $PID $PROGRAM`.

The default is " ".

GUI Equivalent

None

Examples

```
envGetVal("perf" "preReportScript")  
envSetVal("perf" "preReportScript" 'string "~/myCollectSystemInfo.sh")
```

Related Topics

[Reporting Slowness](#)

privateVarAtStartup

```
perf privateVarAtStartup toggle (detect print)
```

Description

Displays all the private environment variables that were modified at Virtuoso startup. The choices are:

- **detect**: Detects all the private environment variables modified at Virtuoso startup (the default)
- **print**: Prints all the information related to the modified private environment variables

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Check*

Field: *Environment Variables*

Examples

The following example returns the current value for the environment variable. The return value of (t nil) indicates that the `detect` option is enabled and the `print` option is disabled:

```
envGetVal("perf" "privateVarAtStartup")  
(t nil)
```

The example below enables `detect` and `print`:

```
envSetVal("perf" "privateVarAtStartup" 'toggle '(t t))
```

Related Topics

[Environment Form](#)

[Diagnostic Center Form](#)

pstackPath

```
perf pstackPath string "check_pstack_path"
```

Description

Specifies the full path to the UNIX *pstack* command. The default is "pstack".

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Health Monitor – Expert – Advanced Debugging – Enable*

Field: *Pstack(Gstack)*

Examples

```
envGetVal("perf" "pstackPath")  
envSetVal("perf" "pstackPath" 'string "/usr/bin/pstack")
```

Related Topics

[Collecting Data Using Health Monitor Tool](#)

stracePath

`perf stracePath string utility_file_path`

Description

Specifies the path to the strace executable file to access the Monitor by Strace form.

The default value is "strace".

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Health Monitor – Advanced Debugging*

Field: *Strace*

Examples

```
envGetVal("perf" "stracePath")  
envSetVal("perf" "stracePath" 'string "/bin/strace")
```

Related Topics

[Health Monitor Form](#)

[Health Monitor \(Advanced\) Form](#)

[Diagnostic Center Form](#)

[sysMonitorPath](#)

sysMonitorPath

`perf sysMonitorPath string utility_file_path`

Description

Specifies the path to the system monitor executable file to access the System Monitor form.

The default value is "ksysguard".

GUI Equivalent

Command: *CIW – Tools – Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced)*

Field: *System Monitor*

Examples

```
envGetVal("perf" "sysMonitorPath")  
envSetVal("perf" "sysMonitorPath" 'string "gnome-system-monitor")
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[Health Monitor Form](#)

[Diagnostic Center Form](#)

[stracePath](#)

sysPulseRefreshInterval

```
perf.hm sysPulseRefreshInterval int time_interval
```

Description

Specifies the interval in seconds to measure the utilization of system resources. You can set the value to 0 to disable the system pulse measurement.

The default is 1, which means to delay the pulse measurement by 1 sec.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *System Pulse – Refresh Interval*

Examples

```
envGetVal("perf.hm" "sysPulseRefreshInterval")  
envSetVal("perf.hm" "sysPulseRefreshInterval" 'int 2)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[sysPulseYellowLightAlert](#)

sysPulseYellowLightAlert

```
perf.hm sysPulseYellowLightAlert int time_interval
```

Description

Specifies the threshold level in seconds of continuous system events to raise a system event alarm.

The default is 3.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *System Pulse – Yellow Light Alert*

Examples

```
envGetVal("perf.hm" "sysPulseYellowLightAlert")  
envSetVal("perf.hm" "sysPulseYellowLightAlert" 'int 30)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[sysPulseRefreshInterval](#)

toolbarAutohide

```
perf.hm toolbarAutohide boolean { t | nil }
```

Description

Automatically hides the Health Monitor when pinned. To use the `toolbarAutohide` environment variable, the `toolbarBorderless` environment variable must be set to `t`.

The default is `nil`.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *Display – Autohide*

Examples

```
envGetVal("perf.hm" "toolbarAutohide")  
envSetVal("perf.hm" "toolbarAutohide" 'boolean t)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[toolbarBorderless](#)

[toolbarNativeWM](#)

toolbarBorderless

```
perf.hm toolbarBorderless boolean { t | nil }
```

Description

Displays the Health Monitor without the form header when pinned.

The default is `t`.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *Display – Borderless*

Examples

```
envGetVal("perf.hm" "toolbarBorderless")  
envSetVal("perf.hm" "toolbarBorderless" 'boolean nil)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[toolbarNativeWM](#)

[toolbarAutohide](#)

toolbarNativeWM

```
perf.hm toolbarNativeWM boolean { t | nil }
```

Description

Enables the Native Window Manager support to resolve the GUI-related issues occurring in the current Window Manager version.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("perf.hm" "toolbarNativeWM")  
envSetVal("perf.hm" "toolbarNativeWM" 'boolean t)
```

Related Topics

[toolbarAutohide](#)

[toolbarBorderless](#)

virtuosoPulseRefreshInterval

```
perf.hm virtuosoPulseRefreshInterval int time_interval
```

Description

Specifies the interval in seconds to trigger a backtrace to get a callstack. You can set the value to 0 to disable the Virtuoso Studio pulse measurement.

The default is 1, which means to delay the pulse measurement by 1 sec.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *Virtuoso Pulse – Refresh Interval*

Examples

```
envGetVal("perf.hm" "virtuosoPulseRefreshInterval")  
envSetVal("perf.hm" "virtuosoPulseRefreshInterval" 'int 3)
```

Related Topics

[Health Monitor \(Advanced\) Form](#)

[virtuosoPulseYellowLightAlert](#)

virtuosoPulseYellowLightAlert

```
perf.hm virtuosoPulseYellowLightAlert int time_interval
```

Description

Specifies the threshold in seconds of consecutive high similarity call chains to report a concentrated pulse.

The default is 5.

The Health Monitor autocorrects the value when the specified integer is invalid.

GUI Equivalent

Command: *Diagnostic Center – Performance – Health Monitor – Health Monitor (Advanced) – Options*

Field: *Virtuoso Pulse – Yellow Light Alert*

Examples

```
envGetVal("perf.hm" "virtuosoPulseYellowLightAlert")  
envSetVal("perf.hm" "virtuosoPulseYellowLightAlert" 'int 30)
```

Related Topics

[sysPulseYellowLightAlert](#)

[sysPulseRefreshInterval](#)

[cpuUtilization](#)

[deleteHMLog](#)

[autoLogLatency](#)

[toolbarBorderless](#)

[toolbarAutohide](#)

User Interface Environment Variables

This section provides information on the names, descriptions, and graphical user interface equivalents for the User Interface environment variables.

accelInput

accelReturnValue

beepVolume

bottomAsstSpansFullWidth

ciwCmdExecuteOnEnter

ciwCmdHistoryInPlace

ciwCmdInputLines

ciwErrorColor

ciwLogHistorySize

ciwMatchCmdColor

ciwMismatchParenColor

ciwOutputWrapMode

ciwRetainUniqueCmds

ciwSyntaxHighlighting

ciwTabStop

ciwWarnColor

dblClkTime

defaultDragColor

defaultEditorBackgroundColor

defaultFloatFieldFormat

typedReturnValue

accelInput

```
ui accelInput boolean { t | nil }
```

Description

Displays accelerated input such as menu and bindkey commands, form and field value settings, and callbacks.

The default is `nil`.

GUI Equivalent

Command: *Options – Log Filter – Set Log File Display Filter*

Form Field: *Accelerated input*

Examples

```
envGetVal("ui" "accelInput")  
envSetVal("ui" "accelInput" 'boolean t)
```

Related Topics

[Changing the Log Filter Options](#)

accelReturnValue

```
ui accelReturnValue boolean { t | nil }
```

Description

Displays return values from accelerated inputs in the output area of the Command Interpreter Window. The default is `nil`.

GUI Equivalent

Command: *Options – Log Filter – Set Log File Display Filter*

Form Field: *Return values from accelerated input*

Examples

```
envGetVal("ui" "accelReturnValue")  
envSetVal("ui" "accelReturnValue" 'boolean t)
```

Related Topics

[Changing the Log Filter Options](#)

beepVolume

```
ui beepVolume int volume
```

Description

Controls the volume level of the system.

The default is 0.

GUI Equivalent

Command: *Options – User Preferences – Command Controls*

Field: *Beep Volume*

Examples

```
envGetVal("ui" "beepVolume")  
envSetVal("ui" "beepVolume" 'int 2)
```

Related Topics

[User Preferences Form](#)

bottomAsstSpansFullWidth

```
ui bottomAsstSpansFullWidth boolean { t | nil }
```

Description

Determines the width of the assistant that is displayed at the bottom of a window. It displays the assistant at the bottom along the entire bottom area of the window.

The default is `nil`, which means that the assistant at the bottom lies between the left and right assistants.

GUI Equivalent

None

Examples

```
envGetVal("ui" "bottomAsstSpansFullWidth")  
envSetVal("ui" "bottomAsstSpansFullWidth" 'boolean t)
```

ciwCmdExecuteOnEnter

```
ui ciwCmdExecuteOnEnter boolean { t | nil }
```

Description

Lets you execute the command specified within the command block.

The default is `t`.

GUI Equivalent

Command: *Options – User Preferences – CIW Controls*

Field: *Enter Key Executes Command*

Examples

```
envGetVal("ui" "ciwCmdExecuteOnEnter")  
envSetVal("ui" "ciwCmdExecuteOnEnter" 'boolean nil)
```

Related Topics

[User Preferences Form](#)

ciwCmdHistoryInPlace

```
ui ciwCmdHistoryInPlace boolean { t | nil }
```

Description

Lets you access history commands within the edit area of the input area using the up and down arrow keys.

The default is `t`.

GUI Equivalent

Command: *Options – User Preferences – CIW Controls*

Field: *History In Place*

Examples

```
envGetVal("ui" "ciwCmdHistoryInPlace")  
envSetVal("ui" "ciwCmdHistoryInPlace" 'boolean nil)
```

Related Topics

[User Preferences Form](#)

ciwCmdInputLines

`ui ciwCmdInputLines int numberOfLines`

Description

Modifies the input area line settings.

The default is 1.

GUI Equivalent

Command: *Options – User Preferences – CIW Controls*

Field: *Input Area Lines*

Examples

```
envGetVal("layout" "ciwCmdInputLines")  
envSetVal("layout" "ciwCmdInputLines" 'int 2)
```

Related Topics

[User Preferences Form](#)

ciwErrorColor

```
ui ciwErrorColor string "color"
```

Description

Specifies the color for error messages that appear in the CIW output area.

The default is "dark red".

GUI Equivalent

None

Examples

```
envGetVal("ui" "ciwErrorColor")  
envSetVal("ui" "ciwErrorColor" 'string "white")
```

ciwLogHistorySize

```
ui ciwLogHistorySize int stopLevel
```

Description

Specifies the maximum number of lines contained in the output area.
The default is 1000.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Output Area Lines*

Examples

```
envGetVal("ui" "ciwLogHistorySize")  
envSetVal("ui" "ciwLogHistorySize" 'int 2)
```

Related Topics

[User Preferences Form](#)

ciwMatchCmdColor

```
ui ciwMatchCmdColor string "color"
```

Description

Specifies the color for command-matching on the input line of the CIW.

GUI Equivalent

None

Examples

```
envGetVal("ui" "ciwMatchCmdColor")  
envSetVal("ui" "ciwMatchCmdColor" 'string "white")
```

ciwMatchParenColor

```
ui ciwMatchParenColor string "color"
```

Description

Specifies the color for parentheses-matching on the input line of the CIW.

GUI Equivalent

None

Examples

```
envGetVal("ui" "ciwMatchParenColor")  
envSetVal("ui" "ciwMatchParenColor" 'string "white")
```

ciwMismatchParenColor

```
ui ciwMismatchParenColor string "color"
```

Description

Specifies the color for parentheses-mismatching on the input line of the CIW.

The default is "red".

GUI Equivalent

None

Examples

```
envGetVal("ui" "ciwMismatchParenColor")  
envSetVal("ui" "ciwMismatchParenColor" 'string "white")
```

Related Topics

[User Preferences Form](#)

ciwOutputWrapMode

```
ui ciwOutputWrapMode cyclic {"Word or Anywhere" | "None" | "Word" | "Anywhere"}
```

Description

Specifies whether to display the CIW output in wrap mode. The default is "Word or Anywhere".

GUI Equivalent

None

Examples

```
envGetVal("layout" "ciwOutputWrapMode")  
envSetVal("layout" "ciwOutputWrapMode" 'cyclic "None")  
envSetVal("layout" "ciwOutputWrapMode" 'cyclic "Word")
```

ciwRetainUniqueCmds

```
ui ciwRetainUniqueCmds boolean { t | nil }
```

Description

Removes identical commands from the input area of the CIW.

The default is `nil`.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Retain Unique Commands*

Examples

```
envGetVal("ui" "ciwRetainUniqueCmds")  
envSetVal("ui" "ciwRetainUniqueCmds" 'boolean t)
```

Related Topics

[User Preferences Form](#)

ciwSyntaxHighlighting

```
ui ciwSyntaxHighlighting boolean { t | nil }
```

Description

Enables checking the syntax or commands entered in the input area of the CIW.

The default is `t`.

GUI Equivalent

Command	<i>Options – User Preferences</i>
Form Field	<i>Syntax Highlighting</i>

Examples

```
envGetVal("ui" "ciwSyntaxHighlighting")  
envSetVal("ui" "ciwSyntaxHighlighting" 'boolean nil)
```

Related Topics

[User Preferences Form](#)

ciwTabStop

`ui ciwTabStop int characterSpaces`

Description

Specifies the number of character spaces that represents a tab in both, the input and output areas of the CIW.

The default is 4.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Tab Stop*

Examples

```
envGetVal("ui" "ciwTabStop")  
envSetVal("ui" "ciwTabStop" 'int 2)
```

Related Topics

[User Preferences Form](#)

ciWarnColor

```
ui ciWarnColor string "color"
```

Description

Specifies the color for warning messages that appear in the CIW output area.

The default is "#eOffe0".

GUI Equivalent

None

Examples

```
envGetVal("ui" "ciWarnColor")  
envSetVal("ui" "ciWarnColor" 'string "#eOffe0")
```

Related Topics

[User Preferences Form](#)

dblClkTime

```
ui dblClkTime int time_interval
```

Description

Sets the interval in milliseconds in which two consecutive mouse clicks are interpreted as a double-click.

The default is 200.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Double Click Time*

Examples

```
envGetVal("ui" "dblClkTime")  
envSetVal("ui" "dblClkTime" 'int 500)
```

Related Topics

[User Preferences Form](#)

defaultDragColor

```
ui defaultDragColor string "color"
```

Description

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.

The default is "yellow".

GUI Equivalent

None

Examples

```
envGetVal("ui" "defaultDragColor")  
envSetVal("ui" "defaultDragColor" 'string "white")  
envSetVal("ui" "defaultDragColor" 'string "darkRed")
```

defaultEditorBackgroundColor

```
ui defaultEditorBackgroundColor string "color"
```

Description

Lets you to change the background color of the editor window. You can reset the background 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.

The default is "black".

GUI Equivalent

None

Examples

```
envGetVal("ui" "defaultEditorBackgroundColor")  
envSetVal("ui" "defaultEditorBackgroundColor" 'string "red")
```

Related Topics

[User Preferences Form](#)

defaultFloatFieldFormat

```
ui defaultFloatFieldFormat string "default_floating_points"
```

Description

Sets the default floating point format for all the hi float fields.

The default is "%.6g".

GUI Equivalent

None

Examples

```
envGetVal("ui" "defaultFloatFieldFormat")  
envSetVal("ui" "defaultFloatFieldFormat" 'string "%.5g")
```

enableFileDialogNameCompletion

```
ui enableFileDialogNameCompletion boolean { t | nil }
```

Description

Controls whether the name auto-complete feature is enabled in the QFileDialog boxes displayed using `hiDisplayFileDialog` and related APIs.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "enableDoubleBuffer")  
envSetVal("ui" "enableDoubleBuffer" 'boolean nil)
```

errorOutput

```
ui errorOutput boolean { t | nil }
```

Description

Displays the error as an output in CIW. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "errorOutput")  
envSetVal("ui" "errorOutput" 'boolean nil)
```

Related Topics

[Customize Toolbar Definition Files](#)

floatPrecision

```
ui.envEditor floatPrecision int maximum_decimal_place
```

Description

Defines the decimal place value greater than or equal to two for any float type variable in Virtuoso. You can define up to the fifteenth decimal place.

The default is 4, which means you can define the decimal place value for the float type variable up to the fourth decimal place.

GUI Equivalent

None

Examples

```
envGetVal("ui.envEditor" "floatPrecision")  
envSetVal("ui.envEditor" "floatPrecision" 'int 5)
```

Related Topics

[Opening Cdsenv Editor in CIW](#)

[Editing an Environment Variable in Cdsenv Editor](#)

focusToFieldSelectsText

```
ui focusToFieldSelectsText boolean { t | nil }
```

Description

Auto selects the content in the text field, when the *Tab* key is pressed to traverse through the editable string fields.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "focusToFieldSelectsText")  
envSetVal("ui" "focusToFieldSelectsText" 'boolean t)
```

Related Topics

[User Preference Options](#)

formDefaultAction

```
ui formDefaultAction string "value"
```

Description

Specifies the *buttonText* of a button to link to the Enter Key.

When this variable is set, the *g_buttonLayout* argument of `hiCreateLayoutForm` is searched for the button whose text equals the value of the variable. The first result found becomes the default button.

If the specified value is not found in *g_buttonLayout*, then Enter performs the same action as the first button of the form button list.

The default is "hiDefault".

GUI Equivalent

None

Examples

```
envGetVal("ui" "formDefaultAction")  
envSetVal("ui" "formDefaultAction" 'string "hiSubmit")
```

Related Topics

[hiCreateLayoutForm](#)

imageTabTip

```
ui imageTabTip boolean { t | nil }
```

Description

Allows you to preview a design by placing the mouse pointer on a tab if multiple designs are open in that tab. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "imageTabTip")  
envSetVal("ui" "imageTabTip" 'boolean t)
```

Related Topics

[Saving Modified Data](#)

interruptCheckInterval

`ui interruptCheckInterval int interval`

Description

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.

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.

GUI Equivalent

None

Examples

```
envGetVal("ui" "interruptCheckInterval")  
envSetVal("ui" "interruptCheckInterval" 'int 200)
```

Related Topics

[User Preferences Form](#)

maximumCopySize

```
ui.text maximumCopySize int maximum_size_limit
```

Description

Modifies the maximum size limit for the `Copy` operation, the operation gets disabled automatically after the data selection exceeds the specified size limit.

GUI Equivalent

None

```
envGetVal("ui.text" "maximumCopySize")  
envSetVal("ui.text" "maximumCopySize" 'int 100000)
```

Related Topics

[Text Device Mode](#)

memoryCheckintervalSeconds

`ui memoryCheckintervalSeconds int time`

Description

Sets the memory check interval from 0 through 60. If the memory check interval is set to 0 then there is no memory checking and if it is set from 1 to 60 then it specifies the number of seconds between each check.

The default value is 1.

GUI Equivalent

Command: *Tools – Voltage Dependent Rules*

Field: *Hierarchy Stop Level*

Examples

```
envGetVal("ui" "memoryCheckintervalSeconds")
envSetVal("ui" "memoryCheckintervalSeconds" 'int 2)
```

Related Topics

[Specifying New Default Values for a Virtuoso Session](#)

mouseMoveSampleRate

`ui mouseMoveSampleRate int mouse_motion_events`

Description

Specifies a percentage (%) of the level of mouse motion events to be processed.
The default value is 100.

GUI Equivalent

None

Examples

```
envGetVal("ui" "mouseMoveSampleRate")  
envSetVal("ui" "mouseMoveSampleRate" 'int 2)
```

Related Topics

[hiSetMouseMoveSampleRate](#)

mouseStopDetectTime

`ui mouseStopDetectTime int time`

Description

Measures the duration of a mouse stopped at a point long enough to classify it as inactive. The default value is 100.

GUI Equivalent

None

Examples

```
envGetVal("ui" "mouseStopDetectTime")
envSetVal("ui" "mouseStopDetectTime" 'int 2)
```

Related Topics

[hiSetMouseStopDetectTime](#)

mouseWheelSpeed

`ui mouseWheelSpeed int speed`

Description

Modifies the 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.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Mouse Prompts*

Examples

```
envGetVal("ui" "mouseWheelSpeed")  
envSetVal("ui" "mouseWheelSpeed" 'int 500)
```

Related Topics

[Specifying New Default Values for a Virtuoso Session](#)

nestLimit

```
ui nestLimit int levelOfCommands
```

Description

Specifies the number of level of commands to be nested. You can select a limit from 1 to 20.

The default is 5.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Nest Limit*

Examples

```
envGetVal("ui" "nestLimit")  
envSetVal("ui" "nestLimit" 'int 8)
```

Related Topics

[User Preferences Form](#)

noWarnOnLVBindKeyCustomization

```
ui noWarnOnLVBindKeyCustomization boolean { t | nil }
```

Description

Disables the warning message when the specified bindkey is not available in the Layout Viewer for customization.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "noWarnOnLVBindKeyCustomization")  
envSetVal("ui" "noWarnOnLVBindKeyCustomization" 'boolean t)
```

Related Topics

[User Preferences Form](#)

openOptionsFormAtMousePos

```
ui openOptionsFormAtMousePos boolean { t | nil }
```

Description

Displays forms that can be opened using the **F3** key at the current mouse pointer position.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "openOptionsFormAtMousePos")  
envSetVal("ui" "openOptionsFormAtMousePos" 'boolean t)
```

Related Topics

[User Preferences Form](#)

optionFormsStayOnTop

```
ui optionFormsStayOnTop boolean { t | nil }
```

Description

Specifies whether the option forms should stay on top of all windows, including the non-Virtuoso windows. The default is `nil`.

To ensure that all option forms follow the same behavior, the environment variable value passed to the first option form issued for all following option forms.

GUI Equivalent

None

Examples

```
envGetVal("ui" "optionFormsStayOnTop")  
envSetVal("ui" "optionFormsStayOnTop" 'boolean t)
```

printAllLVBindKeyBlockWarning

```
ui printAllLVBindKeyBlockWarning boolean { t | nil }
```

Description

Prints the information of customization restriction for each bindkey in Layout Viewer.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "printAllLVBindKeyBlockWarning")  
envSetVal("ui" "printAllLVBindKeyBlockWarning" 'boolean t)
```

Related Topics

[hiSetbindKey](#)

promptOutput

```
ui promptOutput boolean { t | nil }
```

Description

Lets you select the prompt output display on the CIW. The default is `nil`.

GUI Equivalent

Command: *Options – Log Filter – Set Log File Display Filter*

Field: *Prompt output*

Examples

```
envGetVal("ui" "promptOutput")  
envSetVal("ui" "promptOutput" 'boolean t)
```

Related Topics

[Set Log File Display Filter Form](#)

raiseCIWonError

```
ui raiseCIWonError boolean { t | nil }
```

Description

Raises CIW to the front when an error is displayed in the log window. The default is `nil`.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Raise CIW on*

Examples

```
envGetVal("ui" "raiseCIWonError")  
envSetVal("ui" "raiseCIWonError" 'boolean t)
```

Related Topics

[User Preferences Form](#)

[raiseCIWonWarning](#)

raiseCIWonWarning

```
ui raiseCIWonWarning boolean { t | nil }
```

Description

Raises CIW to the front when a warning is displayed in the log window. The default is `nil`.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Raise CIW on*

Examples

```
envGetVal("ui" "raiseCIWonWarning")  
envSetVal("ui" "raiseCIWonWarning" 'boolean t)
```

Related Topics

[User Preferences Form](#)

[raiseCIWonError](#)

releaseBackingStoreOnUnmap

```
ui releaseBackingStoreOnUnmap boolean { t | nil }
```

Description

Allows to release memory when a form or window gets unmapped and restores memory with a redraw when it gets mapped again. It manages memory more effectively, especially when re-use forms are used. You cannot delete re-use forms after they are created.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "releaseBackingStoreOnUnmap")  
envSetVal("ui" "releaseBackingStoreOnUnmap" 'boolean t
```

rememberOptionFormVisibility

```
ui rememberOptionFormVisibility boolean { t | nil }
```

Description

Remembers the option form visibility of a command and applies it when the command is opened again in the same Virtuoso session, when set to `t`.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("ui" "rememberOptionFormVisibility")  
envSetVal("ui" "rememberOptionFormVisibility" 'boolean t)
```

Related Topics

[User Preferences Form](#)

showMouseBar

```
ui showMouseBar cyclic { "Top" | "Bottom" | "None" }
```

Description

Specifies the location of the mouse bindings line in the current window. The `showMouseBar` setting is overridden by any applied workspace regardless of the selected position. The default is `Bottom`.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Mouse Prompts*

Examples

```
envGetVal("ui" "showMouseBar")  
envSetVal("ui" "showMouseBar" 'cyclic "Top")
```

Related Topics

[User Preferences Form](#)

showOptionForms

```
ui showOptionForms boolean { t | nil }
```

Description

Displays the option form for all the commands.

The default is `nil`.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Command Controls*

Examples

```
envGetVal("ui" "showOptionForms")  
envSetVal("ui" "showOptionForms" 'boolean t)
```

Related Topics

[User Preferences Form](#)

sideDockTabs

```
ui sideDockTabs cyclic { "Bottom" | "Side" }
```

Description

Specifies the scope of nets for which labels are to be generated.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Side Dock Tabs*

```
envGetVal("ui" "sideDockTabs ")  
envSetVal("ui" "sideDockTabs " 'cyclic "Side")
```

Related Topics

[User Preferences Form](#)

stopLevel

```
ui.thumbnails stopLevel int stopLevel
```

Description

Specifies the stoplevel of the thumbnail in the cellview window.

The default value is -1.

GUI Equivalent

None

Examples

```
envGetVal("ui.thumbnails" "stopLevel")  
envSetVal("ui.thumbnails" "stopLevel" 'int 1)
```

Related Topics

[Opening a Cellview from the CIW](#)

systemMemoryCheckinterval

`ui systemMemoryCheckinterval int time`

Description

Sets the system memory check interval from 1 to 100. The 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 is going to run in every 20 seconds.

GUI Equivalent

None

Examples

```
envGetVal("ui" "systemMemoryCheckinterval")
envSetVal("ui" "systemMemoryCheckinterval" 'int 5)
```

typedReturnValue

```
ui typedReturnValue boolean { t | nil }
```

Description

Returns the values from typed-in commands in the output area of the Command Interpreter Window.

The default value is `t`.

GUI Equivalent

Command: *Options – Log Filter*

Field: *Return values from typed-in commands*

Examples

```
envGetVal("ui" "typedReturnValue")  
envSetVal("ui" "typedReturnValue" 'boolean nil)
```

Related Topics

[Set Log File Display Filter Form](#)

undoLevel

ui undoLevel int numberOfCommands

Description

Specifies how many commands you can undo with the *Undo* command. This command undoes the most recent commands in reverse of the order in which they were used. You can select a limit from 0 to 128, where 0 disables undo.

The default is 1.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Undo Limit*

Examples

```
envGetVal("ui" "undoLevel")  
envSetVal("ui" "undoLevel" 'int 120)
```

Related Topics

[User Preferences Form](#)

useLineStyleForStrokeText

```
ui useLineStyleForStrokeText boolean { t | nil }
```

Description

Controls if the stroke text in the canvas is drawn with the defined `lineStyle` for the pen assigned to the text.

The default is `nil`, which means that the stroke text is drawn with solid lines irrespective of the `lineStyle` setting.

GUI Equivalent

None

Examples

```
envGetVal("ui" "useLineStyleForStrokeText")  
envSetVal("ui" "useLineStyleForStrokeText" 'boolean t)
```

warningOutput

```
ui warningOutput boolean { t | nil }
```

Description

Generates warning message output in the output area of the Command Interpreter Window.

The default is `t`.

GUI Equivalent

Command: *Options – Log Filter – Set Log Display Filter*

Field: *Warning message output*

Examples

```
envGetVal("layout" "warningOutput")  
envSetVal("layout" "warningOutput" 'boolean nil)
```

Related Topics

[Set Log File Display Filter Form](#)

standardOutput

```
ui standardOutput boolean { t | nil }
```

Description

Selects the standard message output to display on the CIW.

The default is `t`.

GUI Equivalent

Command: *Options – Log Filter – Set Log File Display Filter*

Field: *Standard message output*

Examples

```
envGetVal("ui" "standardOutput")  
envSetVal("ui" "standardOutput" 'boolean nil)
```

Related Topics

[Set Log File Display Filter Form](#)

webBrowser

```
ui webBrowser string "browser_name"
```

Description

Specifies an executable web browser.

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

The default is `firefox`.

GUI Equivalent

Command: *Options – User Preferences*

Field: *Web Browser*

None

Examples

```
envGetVal("ui" "webBrowser")  
envSetVal("ui" "webBrowser" 'string "chrome")
```

Related Topics

[User Preferences Form](#)

Virtuoso Studio Design Environment User Guide

Design Environment Variables

Virtuoso Studio Design Environment Forms

[Add Bookmark Form](#)

[Auto Checkin Preferences Form](#)

[Auto Checkout Preferences Form](#)

[Bindkey Editor Form](#)

[Cdsenv Editor Form](#)

[Close and Purge Data Form](#)

[Close Opened Cellviews Form](#)

[Conversion Toolbox Form](#)

[Data Storage Speed Test Form](#)

[Diagnostic Center Form](#)

[Edit Bookmark Properties Form](#)

[Edit Library Path Form](#)

[Email Tabulated List of Functions Form](#)

[File Preferences Form](#)

[File Preferences Form](#)

[Graphics Performance Benchmarks Window Form](#)

[Health Monitor Form](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

[Health Monitor \(Advanced\) Form](#)

[Load Workspace Form](#)

[Make Read Only Form](#)

[Monitor by Strace Form](#)

[New Hierarchy Form](#)

[New Library Form \(CIW\)](#)

[Open File Form](#)

[Save .cdsenv file Form](#)

[Save As Form](#)

[Save Cellviews Form](#)

[Save Defaults Form](#)

[Save Modified Data Form](#)

[Save Session Form](#)

[Save View Form](#)

[Set Log File Display Filter Form](#)

[Show File Form](#)

[Surveyor: SKILL Tabulator Form](#)

[Software Product License Management Form](#)

[Unable to check out Form](#)

[Unable to check out Form](#)

[User Preferences Form](#)

[ViewFile Window Form](#)

[What's New Search Form](#)

Add Bookmark Form

Use this form to add bookmarks to your session window.

Field	Description
<i>Bookmark</i>	Specifies either to create a new bookmark or add to already existing bookmarks.
<i>Name</i>	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.
<i>Description</i>	Provides a longer description for your bookmark. This text appears in the <i>Description</i> column of the Bookmarks Manager window.
<i>Add to Bookmarks Toolbar</i>	Adds the bookmark to the <i>Bookmarks</i> toolbar.
<i>Bookmark All Tabs</i>	Bookmarks all the open tabs in your session window as a composite bookmark.

Related Topics

[Managing Bookmarks](#)

Auto Checkin Preferences Form

Use this form to set preferences for the automatic check-in feature.

Field	Description
<i>Auto Checkin Preferences</i>	Specifies 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.
<i>When auto checking in properties and files</i>	Specifies whether you want the checkin prompt to appear when you attempt to close and save a file that is not a cellview.
<i>always ask me</i>	Opens a checkin prompt form when 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.
<i>never ask me</i>	Ensures that the checkin prompt form is not displayed when 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.
<i>When auto checking in cellViews</i>	Specifies whether you want the checkin prompt to appear when you attempt to close and save a cellview in a managed library.
<i>never ask me</i>	Ensures that the checkin prompt form is not displayed when you attempt to close and save a cellview in a managed library.
<i>always ask me</i>	Opens a checkin prompt form when you attempt to close and save a cellview in a managed library.
<i>never auto checkin</i>	Ensure that the cellview is not checked in when you attempt to save and close a cellview in a managed library.
<i>always auto checkin</i>	Checks in the cellview when you save and close a cellview in a managed library.

Related Topics

[Options Menu in the CIW](#)

[User Preferences Form](#)

Auto Checkout Preferences Form

Use this form to set preferences for the automatic check-out feature.

Field	Description
<i>Auto Checkout Preferences</i>	Specifies 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.
<i>When auto checking out properties and files</i>	Specifies whether you want the checkout prompt to appear when you attempt to open a file.
<i>always ask me</i>	Opens a checkout prompt form when you attempt to open a file that is not a cellview, such as a property file or data file, in a managed library.
<i>never ask me</i>	Ensures that the checkout prompt form is not displayed when you attempt to open a file that is not a cellview, such as a property file or data file, in a managed library.
<i>When auto checking out cellViews</i>	Specifies whether you want the checkout prompt to appear when you attempt to open a cellview in a managed library.
<i>always ask me</i>	Opens a checkout prompt form when you attempt to open a cellview in a managed library.
<i>never ask me</i>	Ensures that the checkout prompt form is not displayed when you attempt to open a cellview in a managed library.

Related Topics

[Options Menu in the CIW](#)

[User Preferences Form](#)

Benchmark Results Window Form

Displays the benchmark test results generated by the Graphics Performance Benchmarks window.

Field	Description
<i>Name</i>	Lists 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 <i>Options</i> menu.
<i>Weight</i>	Displays the weight of an individual benchmark that is used to measure its contribution to the performance of its benchmark group.
<i>Performance Chart</i>	<p>Displays the 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.</p> <p>The color schemes <i>Green</i>, <i>Yellow</i>, and <i>Red</i> are used to relay <i>Performance Chart</i> results.</p> <p>Additionally, you can use the <i>Options</i> menu to change the results display from a <i>2D Bar</i> to a <i>3D Bar Chart Presentation</i>, and also to choose whether to display <i>All Benchmarks</i> or <i>Only Selected Benchmarks</i>.</p>
<i>Your Speed (/s)</i>	Displays the absolute performance for a benchmark test.
<i>Ref Speed (/s)</i>	Displays the collective performance that represents a reference standard for a smooth performance.
<i>%</i>	Displays the ratio of <i>Your Speed</i> against <i>Ref Speed</i> .

Related Topics

[Graphics Performance Benchmarks Window Form](#)

Bindkey Editor Form

Use this form to configure the bindkey settings for an application.

Field	Description
<i>Application Tree</i>	Displays applications for which bindkeys are currently defined.
<i>Show applications with no bindkeys defined</i>	Displays the applications that do not currently have their bindkeys defined.
<i>Save Bindings For</i>	Selects which applications to be taken into consideration by the <i>Preview</i> and <i>Save</i> operations.
<i>Preview</i>	Select to display a text window detailing the bindkeys. Applications that inherit the bindkeys for the currently selected application are also listed.
<i>Load</i>	Displays the Load Bindkeys form where you can load previously saved bindkeys for use.
<i>Save</i>	Selects to display the Save Bindkeys form where you can save bindkey settings. Bindkeys are not autoloaded. You must explicitly load any saved bindkey files from a file which is automatically loaded, such as the <code>.cdsinit</code> file.
<i>Search</i>	Specifies search criteria to filter the bindkeys that are listed in the <i>Bindkey Table</i> . You can use the <i>Search</i> field to group bindkeys by the same base key. For example, if you wanted to show all bindkeys that are associated with the <code>A</code> key, enter <code><KEY>A</code> in the <i>Search</i> field. This displays all the associated <code>A</code> key bindkeys, for example <code>Shift</code> , <code>Ctrl</code> , and <code>Ctrl+Shift</code> .
<i>Bindkey Table</i>	Lists all of the current bindkeys associated with the currently selected application(s) in the <i>Application Tree</i> , including their related <i>Command</i> and <i>EnterFunction Command</i> .
<i>Bindkey</i>	The current key or mouse binding for the selected bindkey.
<i>Command</i>	Displays the SKILL function associated with the currently selected bindkey.
<i>EnterFunction Command</i>	Displays the user entry function (also called an “enterfunction”) that is currently associated with the selected bindkey.



Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Show</i>	<p>Shows or hides the following types of bindkeys:</p> <ul style="list-style-type: none">■ <i>Inherited</i>■ <i>Blocked</i>■ <i>Empty</i> <p>Based on your selection, Bindkey Editor appropriately highlights bindkeys. Bindkeys that have been overridden, using <code>nil</code>, in a child application are represented in the child application using strikethrough text.</p> <p>By default, the <i>Inherited</i> 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.</p>
<i>Block Binding and Unblock Binding</i>	<p>Blocks or unblocks binding the selected bindkey. Use block option to prevent propagation to the parent application. On the other hand, use unblock to allow propagation to the parent application.</p> <p>These options are available for bindkeys of a child application.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Application</i>	<p>Shows the application that the bindkey is associated with if more than one application is selected in the <i>Application Tree</i>. Otherwise, the column does not get displayed.</p> <p>Displaying the related application is especially useful in the following situations:</p> <ul style="list-style-type: none">■ When a child application is selected and <i>Show inherited bindings</i> is checked. <p>Here, multiple applications can be listed in the <i>BindKey Table</i>, and the <i>Application</i> column header gets sorted by application; distinguishing inherited bindings from those bindings defined for the current application.</p> ■ When multiple applications are selected in the <i>Application Tree</i>. <p>Here, for example, you may now want to use the <i>BindKey Table</i> to view bindings for <code>Ctrl<Key>X</code> across all applications (to achieve this, <code>shift-select</code> the applications in the <i>Application Tree</i>, then use <i>Search</i> to filter the bindings). Without the <i>Application</i> column you would not know which application a bindkey was inherited from.</p>
<i>Add binding</i> 	Creates a new bindkey.
<i>Remove binding</i> 	Removes an existing (customized) bindkey.

Related Topics

[Configuring Application Bindkeys](#)

[Viewing the Current Bindkeys for an Application](#)

[Restrictions to Setting Bindkeys](#)

[Format for Key and Mouse Bindings](#)








Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Determining SKILL Function for a Menu Command

Cdsenv Editor Form

Use this form to search for environment variables across multiple `.cdsenv` files associated with different tools. This form allows you to navigate to a specific environment variable and update it as needed.

Field	Description
<i>File</i>	<p>Specify the following options:</p> <ul style="list-style-type: none"> ■ <i>Save</i> Displays the Save <code>.cdsenv</code> file form that allows you to customize how updated environment variables are saved in a <code>.cdsenv</code> file. ■ <i>Load File As Overlay</i> Displays the Load <code>.cdsenv</code> file dialog box that allows you to load the selected <code>.cdsenv</code> file to overlay the loaded environment variables. ■ <i>Quit</i> Closes the Cdsenv Editor form.
<i>View</i>	<p>Show/Hide the <i>Loaded From File</i> field in the Cdsenv Editor table.</p> <p> Displays the Save <code>.cdsenv</code> file form.</p> <p> Displays the Load <code>.cdsenv</code> file dialog box that allows you to load the selected <code>.cdsenv</code> file.</p> <p> Expands all tools in the Cdsenv Editor table.</p> <p> Collapses all tools in the Cdsenv Editor table.</p> <p> Searches for the specified environment variable across all <code>.cdsenv</code> files.</p>
<i>Tool</i>	<p>Displays the list of tool names that allows you to navigate and display all environment variables for a tool.</p> <p>The folder icons are displayed as follows:</p> <ul style="list-style-type: none"> ■  : Indicates the environment variables for the selected tool are loaded for the current Virtuoso session. ■  : Indicates the environment variables for the selected tool are not loaded for the current Virtuoso session.
<i>Variable</i>	Contains the names of the environment variables.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Type</i>	Specifies the type of environment variables. For example, boolean, string, cyclic, and int.
<i>Value</i>	Specifies the current value of an environment variable. You can edit the specified value of an environment variable from here.
<i>Status</i>	<p>Shows the current status of the variables.</p> <p>Valid values:</p> <ul style="list-style-type: none">■ <i>default</i>: No value of an environment variable gets modified.■ <i>Modified</i>: The value of an environment variable gets modified but remains the same as default.■ <i>modified from default</i>: The value of an environment variable gets modified to a different one from default.
<i>Loaded From File</i>	Shows the file from which the selected <code>.cdsenv</code> file has been loaded. It only shows the path that is not default Cadence installation location.

Related Topics

[Opening Cdsenv Editor in CIW](#)

Design Form

Use this form to check library-related information in the CIW. You can use the options on this form to run tests on the specified design. To reduce the size of the design on disk, you need to save, close, reopen, and save the design again.

Field	Description
<i>Library</i>	Specifies the library that contains the design that you want check. Click <i>Browse</i> to select the library from Library Browser.
<i>Cell</i>	Specifies the cell name.
<i>View</i>	Specifies the view name.
<i>Check</i>	Selects the tests you want to run on the specified design.
<i>Recursion</i>	Check the design hierarchy for recursive references (loops) that can cause a tool to become unresponsive. These references must be removed.
<i>Design statistics and memory usage</i>	Checks the design statistics and reports the amount of memory used by the design. <ul style="list-style-type: none">■ <i>Full design hierarchy</i>: Runs the check on the entire design hierarchy.
<i>Duplicate instances and shapes</i>	Checks the design to identify any duplicate instances or shapes in it. <ul style="list-style-type: none">■ <i>Create markers</i>: Creates markers to indicate the location of the duplicate instances or shapes in the design. Use the Annotation Browser to view the markers.■ <i>Remove duplicates</i>: Deletes any duplicate instances or shapes detected during the check.■ <i>Full design hierarchy</i>: Runs the check on the entire design hierarchy.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>OA Integrity (oaScan)</i>	<p>Runs the oaScan utility that checks for inconsistencies in the OpenAccess design, technology, and DM databases, and optionally repairs the inconsistencies and saves the databases.</p> <ul style="list-style-type: none">■ <i>Repair</i>: Repairs any issues found during the oaScan.■ <i>Full design hierarchy</i>: Runs the check on the entire design hierarchy. <p>To repair the full design hierarchy, use the <i>Scan Hierarchy</i> option from the Data Integrity tab of the Diagnostic Center.</p>
<i>Constraint Integrity</i>	<p>Checks for duplicate constraint members, unreferenced constraints, and invalid constraints and optionally removes them from the design.</p> <ul style="list-style-type: none">■ <i>Remove duplicate group members</i>: Removes duplicate constraint members and displays progress and duration to complete the process.■ <i>Remove unreferenced constraints</i>: Removes unreferenced constraints that are not a part of any constraint group.■ <i>Remove invalid constraints</i>: Removes invalid constraints that do not have any members.■ <i>Remove out-of-context constraints and members</i>: Removes all out-of-context members and the associated constraints. Constraints and constraint members become out of context when the design objects they refer to are removed from the design, for example, after deleting instances, nets, or pins.

Related Topics

[oaScan Utilities for Virtuoso](#)

[Scan/Repair Hierarchy Form](#)

[Diagnostic Center Form](#)

[Environment Form](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Library Form

SKILL Form

Check Environment Variables Form

Environment Form

Use this form to run various diagnostic tests.

Field	Description
<i>Environment Variables</i>	Displays the Check Environment Variables form. This form provides options to report issues caused by the Virtuoso and UNIX environment variables with modified values.
<i>Cadence Generic DM</i>	Displays the Test GDM form. In the form, provide the <i>Library</i> , <i>Cell</i> , and <i>View</i> information for a DM-managed cellview. This test helps in checking the performance of your DM system. It measures the real, user, and system time to get its current status.
<i>Graphics Benchmark</i>	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.
<i>Remote Desktop</i>	Displays the Check Remote Desktop form. Use this form to measure the network latency from the remote desktop to your client system.
<i>License</i>	Displays the <i>Performance Tests</i> tab on the Software Product License Management form.
<i>System</i>	Displays a text window that provides information regarding your system.
<i>Data Storage Speed Test</i>	Opens the Data Storage Speed Test Form to run the data storage speed test for a specified path.

Related Topics

[privateVarAtStartup](#)

[Measuring Graphics Performance](#)

[Data Storage Speed Test Form](#)

[Diagnostic Center Form](#)

[Design Form](#)

[Library Form](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

SKILL Form

Check Environment Variables Form

Library Form

Use this form to check the design library access time with multiple pings.

Field	Description
<i>Library Information</i>	Opens a log file containing all the library-related information, such as <code>cdsinfo</code> , cell count, and permissions.
<i>Library Diagnostic</i>	Scans all libraries and disk mount points for network latency. You can also access the same function from <i>Library Manager – Help – Diagnostics</i> .

Related Topics

[Diagnostic Center Form](#)

[Design Form](#)

[Environment Form](#)

[SKILL Form](#)

[Check Environment Variables Form](#)

SKILL Form

Use this form to perform SKILL profiling.

Field	Description
<i>Profiler</i>	Opens the Profiler assistant in SKILL IDE to check the time and memory consumption of the SKILL commands.
<i>Profiler Summary</i>	Displays a profiler summary.
<i>Replay</i>	Reads a replay file from a <code>cdsmps</code> control console.
<i>Other Tips</i>	Provides useful tips to debug performance issues in Virtuoso.

Related Topics

[Diagnostic Center Form](#)

[Design Form](#)

[Environment Form](#)

[Library Form](#)

[Check Environment Variables Form](#)

[Checking the Profiler Summary](#)

[Controlling the SKILL Replay](#)

Check Environment Variables Form

Use this form to check the information related to Virtuoso and Unix environment variables.

Field	Description
Report Environment Variables Affecting Performance	Reports the environment variables that are affecting the performance of the current session.
<i>Modified Virtuoso environment variables</i>	Specifies whether to report the modified Virtuoso environment variables that are affecting the performance of the current session.
<i>Unix environment variables and settings</i>	Specifies whether to report the modified Unix environment variables and settings that are affecting the performance of the current session.
<i>Tool to Check</i>	Specifies the tools to check. The options are: <ul style="list-style-type: none">■ <i>All covered tools</i>: Checks the tools listed in the <i>Covered Tools</i> section.■ <i>All loaded tools</i>: Checks the tools listed in the <i>Currently Loaded</i> section.■ <i>Loaded tools by name (fill in Tool Names)</i>: Checks the tools specified in the Tool Names section.
<i>Covered Tools</i>	Lists all the covered tools.
<i>Currently Loaded</i>	Lists all the currently loaded tools.
<i>Tool Names</i>	Specifies the tool name to check the performance. This option is enabled, when <i>Loaded tools by name (fill in Tool Names)</i> is selected.
<i>Windows to Check</i>	Checks the layout window. The options are: <ul style="list-style-type: none">■ <i>All layout windows</i>: Checks all the available layout windows.■ <i>Current layout windows</i>: Checks the currently opened layout window.
View Modified Virtuoso Environment Variables	Displays all the modified environment variables.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Public</i>	Displays all the modified environment variables in the Cdsenv Editor.
<i>Private</i>	Displays all the private environment variables that were modified at the Virtuoso startup.

Related Topics

[Design Form](#)

[Environment Form](#)

[Library Form](#)

[SKILL Form](#)

[Diagnostic Center Form](#)

Close and Purge Data Form

Use this form to close and delete data from virtual memory.

Field	Description
<i>All</i>	Selects all the cellviews in all the listed libraries.
<i>None</i>	Selects any selected cellviews.
<i>For all modified data</i>	Sets how Virtuoso handles modified data on <i>File – Close Data</i> selection.
<i>prompt</i>	Prompts to always save the data on close.
<i>save</i>	Saves data automatically on closing the window.
<i>discard</i>	Discards unsaved data if the Virtuoso Studio Design Environment is closed using <i>File – Close Data</i> .
<i>Library</i>	Displays the names of the libraries containing the cellviews that you had opened. You can close and delete their data from virtual memory.
<i>Cell/Library File</i>	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.
<i>View/Cell File</i>	Displays the name of the cellview, for example, <code>schematic</code> , <code>symbol</code> , and so on that you have opened for viewing or editing. You can close and delete their cellview or file data from virtual memory.
<i>Mode</i>	Displays whether the opened cellview is in read-only mode or in edit-only mode.

Related Topics

[Closing Cellviews in Virtuoso](#)

[dbUndoAcrossPurge](#)

Close Opened Cellviews Form

Use this form to view the currently open cellviews and allows you to save and close them.

Field	Description
<i>Library</i>	Lists the name of the library for which you have opened cellviews for viewing or editing.
<i>Cell</i>	Displays the name of all the cells corresponding to the library that you have opened for viewing or editing.
<i>View</i>	Displays the names of the cellviews that you have opened.
<i>Status</i>	Displays the status of the opened cellviews. If the cellviews have not changed, the <i>Status</i> field remains blank. If the cellviews have been changed, the <i>Status</i> field displays <i>modified</i> .
<i>Save modified cellviews</i>	Allows you to save the opened cellviews before defragmentation. All modified cellviews are saved by default.

Related Topics

[Closing Cellviews in Virtuoso](#)

Conversion Toolbox Form

Use this form to convert different types of design data from one format or organization to another. You can use the Conversion Toolbox to create a non-proprietary snap-shot of your SKILL environment, and convert your schematic libraries and designs.

Field	Description
<i>SKILL Surveyor</i>	Opens the SKILL Tabulated Form to create a non-proprietary snap-shot of your SKILL environment and report the Cadence functions called from your SKILL code.
<i>Check SKILL Name Conflicts</i>	Opens the SKILL Name Checker form and lets you select the conflicts to run the check.
<i>Convert Libraries to Use physConfigs</i>	Opens the Convert Libraries to Use Physical Configuration Views form so that you can convert your schematic libraries and designs.
<i>XOR layers between a SiP file and OA layout</i>	Opens the XOR SiP against OA form and compares a SiP file against an OpenAccess layout.
<i>Convert DE-HDL libraries to Unified libraries</i>	Opens the Convert DE-HDL libraries to Unified libraries form.

Related Topics

[Surveyor: SKILL Tabulator Form](#)

[Convert DE-HDL libraries to Unified libraries](#)

[SKILL Name Checker](#)

[XOR SiP against OA Form](#)

[Convert Libraries to Use Physical Configuration Views Form](#)

Convert DE-HDL libraries to Unified libraries

Use this form to convert design entry HDL library to unified libraries.

Field	Description
<i>Library</i>	Specifies the name of the library.
<i>Part File</i>	Lets you choose a Part table file (.ptf).
<i>Run directory</i>	Specify the path of the directory.

Related Topics

[Conversion Toolbox Form](#)

[Surveyor: SKILL Tabulator Form](#)

Data Storage Speed Test Form

Use this form to test the data storage speed of a specified path.

Field	Description
<i>Test Path</i>	Specifies the test path for which the data storage speed test needs to be done.
<i>Disk I/O (dd)</i>	Reports disk bandwidth using the <code>dd</code> command.
<i>Sequential read/write</i>	Measures disk bandwidth by creating a file that contains null characters using the <code>dd</code> command.
<i>File Size (MB)</i>	Specifies the test file block size of the <code>dd</code> command. For example, if you specify <i>File Size (MB)= 1</i> the following command is executed to report disk bandwidth: <pre>dd if=/dev/zero of=./test1 bs=1M count=1</pre>
<i>Database Query</i>	Reports database query transition time.
<i>SQLite</i>	Reports SQLite database query transition time.
<i>Number of Records</i>	Specifies the number of SQL table records created for testing.
<i>Number of Queries</i>	Specifies the number of database queries.
<i>File Status Query (stat)</i>	Measures the performance of the file status query using a <code>stat</code> system call and reports the time per <code>stat</code> call.
<i>Number of Calls</i>	Specifies the number of <code>stat</code> calls.
<i>Benchmark Control</i>	Measures performance in regular intervals.
<i>Iterations</i>	Specifies the number of iterations to be measured.
<i>Interval (sec)</i>	Specifies the time to wait before measuring the next iteration.

Related Topics

[Environment Form](#)

Diagnostic Center Form

Use this form to troubleshoot any Virtuoso or non-Virtuoso related issues in the current session.

Tab	Description
<u>Performance</u>	Contains various debugging techniques to detect issues that might have caused the current session to slow down or freeze.
<u>Data Integrity</u>	Checks for inconsistencies in the contents of a library or cellview, and optionally repairs them.

Performance

The following table describes the fields available on the *Performance* tab of the Diagnostic Center form.

Field	Description
<u>Health Monitor</u>	Monitors the performance and activities of Virtuoso, with an option to record callstacks for reporting the issues.
<u>Check</u>	Performs various checks to search for the root cause affecting the performance of the current session.
<u>Library</u>	Checks the design library access time with multiple pings.
<u>Design</u>	Checks specified designs or all designs in a library for better performance.
<u>Environment</u>	Checks system environment or environment variables for better performance.
<u>SKILL</u>	Performs SKILL profiling.

Data Integrity

The following table describes the fields available on the *Data Integrity* tab of the Diagnostic Center form.

Field	Description
<i>Check</i>	Checks for inconsistencies in the contents of a library or cellview, and optionally repairs them.
<i><u>Layout Integrity</u></i>	Checks for layout integrity issues, such as missing instance masters or via definitions.
<i>Constraints Integrity</i>	Checks for constraint integrity issues, such as invalid or unreferenced constraints.
<i>Database Integrity</i>	Checks for data integrity issues using the oaScan features. It provides two options: <ul style="list-style-type: none">■ <i><u>Scan Library</u></i>■ <i><u>Scan Hierarchy</u></i>

Related Topics

[Diagnostic Center Overview](#)

[Health Monitor Overview](#)

[Enabling the oaScan Utilities and Specifying the oaScan Version to Use](#)

[Collecting Data Using Health Monitor Tool](#)

[Troubleshooting an Unresponsive Virtuoso Application](#)

Edit Bookmark Properties Form

Use this form to edit bookmark properties.

Field	Description
<i>Name</i>	Displays the name of the current design cellview or cellviews. You can type a different name in this field.
<i>Description</i>	Specifies a description for your bookmark. This text appears in the <i>Description</i> column of the Bookmarks Manager window.
<i>Add to Bookmarks Toolbar</i>	Adds the bookmark to the <i>Bookmarks</i> toolbar.

Related Topics

[Add Bookmark Form](#)

[Managing Bookmarks](#)

Edit Library Path Form

Use this form to modify the library path.

Field	Description
<i>Library</i>	Specifies a library name and path to edit.
<i>Name</i>	Specifies name to the library.
<i>Directory</i>	Selects the directory to which you assign the library if you are not using design management.
<i>Design Manager</i>	Specifies the design management setup. If you are using design management, you must put the library in the design-managed workarea from which you started the software.
<i>Cadence DM</i>	Opens your library under design management.
<i>No DM</i>	Provides no design management features, such as check in and check out, for your library.

Related Topics

[Add Library Form](#)

Email Tabulated List of Functions Form

Use this form to locate a valid mail command on your system (`mail` or `mailx` in `/usr/bin` or `/bin`) after you have provided all the details in the SKILL Tabulator Form.

Field	Description
<i>Email tabulated list of functions to Cadence for analysis</i>	Specifies whether to send the output report file. The default is <code>./skillTab.out</code> to Cadence Support for analysis.
<i>Your email address to receive analysis results</i>	Specifies the e-mail address that Cadence should use to send the analysis results back to you. By default, the e-mail address you type in the <i>Customer Email Address</i> field on the SKILL Tabulator form appears in this field.
<i>Your Name</i>	Specifies the name of the user.
<i>Your Company Name</i>	Specifies the name of the user company.
<i>Your Company Location</i>	Specifies the address of the user company.

Related Topics

[Surveyor: SKILL Tabulator Form](#)

[Operating SKILL Surveyor](#)

File Preferences Form

Use this form to set the file preferences for the Library Browser.

Field	Description
File Preference	Specifies the maximum number of entries that can appear on the recently used file list.
<i>Recently used file list</i>	Specifies the maximum number of files you want to see on this list. The maximum is 20. The program applies your settings immediately, you do not have to restart the software.
<i>Auto Save Database After</i>	Specifies the number of database edits required before an auto save is performed. The default is 500. The minimum acceptable edit value is 100, and the maximum is 9999.
Open Browser Automatically For	Decides which forms you want the Library Browser to open up with.
<i>Open Form</i>	Selects if you want the Library Browser to open with the Open File form. <ul style="list-style-type: none"> ■ <i>yes</i>: Opens the Library Browser automatically, along with the Open File form, when you choose File – Open from the CIW. ■ <i>no</i>: No change takes place.
<i>Create Instance Forms</i>	Selects if you want the Library Browser to open with the Create Instance forms. <ul style="list-style-type: none"> ■ <i>yes</i>: Opens the Library Browser automatically when you choose <i>Add – Instance</i> or <i>Create – Instance</i> from a schematic or layout window. ■ <i>no</i>: No change takes place.
<i>Other Forms</i>	Selects if you want the Library Browser to open with the other commands set in the <code>.cshrc</code> file. <ul style="list-style-type: none"> ■ <i>yes</i>: Opens the Library Browser automatically when you select other commands you set in your <code>.cshrc</code> file. ■ <i>no</i>: No change takes place.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>CIW Preferences</i>	Sets the display preferences for CIW.
<i>Prompt On Exit</i>	Selects if you want the Exit dialog box to get displayed automatically when exiting the CIW. <ul style="list-style-type: none">■ <i>yes</i>: Displays the Exit dialog box automatically each time you choose <i>File – Exit</i> from the CIW.■ <i>no</i>: No change takes place.

Related Topics

[Options Menu in the CIW](#)

[User Preferences Form](#)

Graphics Performance Benchmarks Window Form

Use this window to select the benchmark test you want to perform. The Graphics Performance Benchmarks window is displayed when you run the `hiGraphicsBenchmark` command from a shell terminal.

Field	Description
File menu	Performs various actions on the Graphics Performance window.
<i>Run Selected Benchmark</i>	Runs the benchmark tests that are currently selected in the <i>Benchmarks</i> table. This is equivalent to using the <i>Run</i> button.
<i>View Saved Results</i>	Displays the Open form from where you can open and view previously saved results.
<i>Tools menu</i>	Choose the appropriate command to open the Task Viewer window or to run the Calibrator.
Options	Specifies the controls of the benchmark test.
<i>Selections</i>	Filters the benchmark test to be performed. For example, select <i>All</i> tests, or only those tests that are part of the <i>Redraw Set</i> .
<i>Repeats</i>	Specifies how many times the selected tests must be run. For example, if you have selected the <i>REDRAW ON WINDOW</i> group, and have set <i>Repeats</i> to <i>Twice</i> (the default), the test group then runs twice. Using <i>Repeats</i> can increase testing accuracy because the program keeps track of the number of times the tests are run and reports an average result.
<i>Duration</i>	Specifies the time for which each test must run. For example, if you set the <i>Duration</i> 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

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Testing Window</i>	<p>Specifies the display settings of the window. The default is to display the window <i>With Frame</i>.</p> <ul style="list-style-type: none">■ 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 also gets 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.
<i>Remember Settings</i>	<p>Saves the latest settings to the <code>.benchmark</code> file in your home directory. Saved values are used as the default values the next time the program is run</p>
<i>Benchmarks table</i>	<p>Displays information about all the available graphics performance tests (<i>Name</i> column) that can be run. Select the check box corresponding to a test if you want to run that test.</p> <p>Tests are grouped into specific testing areas, such as <i>REDRAW ON WINDOW</i>, <i>AREA COPY</i>, and <i>IMAGE OPERATION</i>.</p> <p>A <i>Weight</i> and <i>Description</i> can also be provided for each benchmark test.</p> <p>To edit the weight (importance) to be applied to a test, double-click the value in the <i>Weight</i> column and type the new value, or use the up and down arrows.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

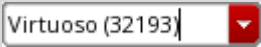




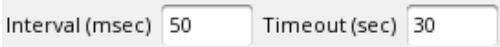
Field	Description
<i>Run</i>	<p>Runs the selected tests.</p> <p>The <i>Run</i> button becomes available only if one or more specific benchmark tests are selected in the <i>Benchmarks Table</i>.</p> <p>Note: After testing starts, the <i>Run</i> button changes to a <i>Stop</i> button to enable you to interrupt the testing, if required.</p>

Related Topics

[Benchmark Results Window Form](#)

Health Monitor Form

Use this form to control the data collection when any Virtuoso application exits unexpectedly or stops responding. This data is then transferred to the support team to resolve the issue.

Field	Description
Process ID 	This section provides options to specify the process ID to track its performance.
Refresh 	Refreshes the <i>Process ID</i> drop down list.
Copy 	Copies the process ID to the clipboard. This ID can be used in conjunction with UNIX commands in <i>Expert</i> mode.
Search 	Selects a frozen application window to bring up the Health Monitor form with the process ID of this window.
Pin 	Pins the minimized version of the form at the top of the screen.
Program Execution Tracing	This section provides options to specify the time for recording the performance of the selected process ID.
Recording Time	<p>Specifies the time for which you want to collect the callstacks for the specified <i>Process ID</i>. Move the slider to the specified time on the <i>Recording Time</i> bar.</p> <p>If you move the slider to <i>Custom</i>, additional options are displayed to let you specify the <i>Interval</i> and <i>Time out</i> information.</p> 
Latency (sec)	Specifies 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	<p>Specifies the keywords to be appended to the log file name for categorization.</p> <p>Environment Variable: <u>fileTag</u></p>









Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
Ready <input type="checkbox"/> Ready	<p>Attaches GDB and prepares the tracker to capture minor slowness issues before recording the callstacks. This option is visible only when <i>Print all threads</i> is enabled or the value in <i>Process ID</i> is set for a non-Virtuoso process.</p>
Start Recording / Stop Recording	<p>Records the callstacks to collect the performance data of the specified process ID. The default recording time is 180 seconds.</p> <p>You can check the remaining recording time in the progress bar available next to the <i>Start Recording</i> button or stop recording when the issue disappears.</p>
Tool Appears Frozen	<p>This section provides options to terminate a frozen tool or application.</p>
Terminate Virtuoso	<p>Displays the Terminate Virtuoso form.</p> <p>Click the <i>Send CTRL-C</i> button to interrupt the program causing the issue. If it is still unresponsive, specify the reason for exiting Virtuoso (for example, describe how you ran into this issue) and then click the <i>Terminate</i> button.</p> <p>The Performance Diagnostic tool records 10 seconds of callstacks, if possible, generates a <code>hangReport</code>, and then closes the current Virtuoso session.</p>
Expert	<p>Displays the <i>Advanced Debugging</i> section.</p> <p>Environment Variable: <code>enableExpertMode</code></p>
Advanced Debugging	<p>This section provides a UNIX expert with more options to apply advanced debugging techniques.</p>
Enable	<p>Enables the advanced debugging options.</p>
Pstack (Gstack)	<p>Invokes the <code>pstack</code> or <code>gstack</code> UNIX command to print a callstack of the specified PID</p> <p>Environment Variable: <code>pstackPath</code></p>
Strace	<p>Opens the Monitor by Strace form, where you can specify the data analysis mode to trace system calls of the specified process ID.</p> <p>Environment Variable: <code>stracePath</code></p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Debugger (gdb)</i>	Invokes the <code>gdb</code> UNIX command to attach a GNU Debugger to the specified process ID. Environment Variable: <code>gdbPath</code>
<i>Open Terminal</i>	Opens the UNIX shell terminal window.
<i>Status Indicator</i>	Indicates the health status of your current session. These are the indicators: <ul style="list-style-type: none"> ■ Off : Indicates the Health Monitor is off. Click to activate the Health Monitor. ■ Idle : Indicates the Health Monitor is on and the current Virtuoso session is idle. ■ Green : Indicates the pulse is vibrant, which means the current Virtuoso session is busy but running, and the system resources are allocated as expected. ■ Yellow : Indicates that the pulse is concentrated, which means Virtuoso is busy running a specific sub-feature or the system resource usage is high. ■ Red : Indicates the pulse is low, which means Virtuoso is calling the same leaf function or the majority of the system resources are occupied. ■ Flash  : Indicates that the status change is unlikely to have been caused by Virtuoso and is related to other factors outside the Virtuoso environment.
<i>Health Monitor (Advanced)</i>	Opens the Health Monitor (Advanced) form.
	
<i>Entry command</i>	Displays the user command for the current session.
<i>Call chain</i>	Displays the calling relationship of the software packages. A call chain is an abstract of a callstack. This information helps understand the purpose of the program.

Related Topics

[Monitor by Strace Form](#)

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Health Monitor Overview

Health Monitor (Advanced) Form

Use this form to get a detailed performance report of your Virtuoso session and to view various scripts at different intervals to detect non-virtuoso issues, plot charts, and query historical system statistics.

Tab	Description
<u>Main</u>	Displays all the basic information about the current Virtuoso session and the machine.
<u>Log</u>	Displays the information related to log files.
<u>System</u>	Displays all the system-related information.
<u>Options</u>	Provides options to update the Health Monitor settings.

Command	Description
<i>Start/Stop</i>	When <i>Start</i> is selected, the Health Monitor is activated.
<i>Freeze/Unfreeze</i>	When <i>Freeze</i> is selected, the form freezes at that particular interval. When <i>Unfreeze</i> is selected, it resumes refreshing all the fields on the form.
<i>Capture</i>	Captures the pulse and generates a log file with all the data.
<i>System Monitor</i>	Launches the System Monitor to monitor information and statistics about the system.

Main

The following table describes the fields available on the *Main* tab of the Health Monitor Advanced form.

Field	Description
<i>Description</i>	Displays information related to the entry command, call chains, and errors.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Procedure/Machine Information</i>	<p>Displays all the machine related information. The following fields are listed:</p> <ul style="list-style-type: none">■ State: Displays the current state of the Virtuoso process. The following status is displayed:<ul style="list-style-type: none"><input type="checkbox"/> R(running)<input type="checkbox"/> S(sleeping)<input type="checkbox"/> D(disk sleep)<input type="checkbox"/> T(stopped)<input type="checkbox"/> t(tracing stop)<input type="checkbox"/> Z(zombie)<input type="checkbox"/> X(dead)■ VmPeak: Displays peak virtual memory size.■ VmSize: Displays the virtual memory size.■ Threads: Displays the number of threads used in the process.■ loadavg: Displays the system load average figures.
<i>Runtime Callstack</i>	Displays the present callstack of the program.

Log

The following table describes the fields available on the *Log* tab of the Health Monitor Advanced form.

Field	Description
<i>Log Files</i>	Displays the location of log files.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Last 100 lines of CDS.log</i>	<p>Displays the last 100 lines of <code>CDS.log</code>.</p> <p>In addition to <code>CDS.log</code>, pulse event logs are generated automatically. A pulse event log includes information for the begin/end timing and a link to extra event files containing all details, for example, pulse type, brief description, call stack history, and the analysis result.</p>

System

The following table describes the fields available on the *System* tab of the Health Monitor Advanced form.

Field	Description
<i>CPU</i>	<p>Provides graphical representation based on time and utilization (%) to calculate the CPU performance. Displays the following information:</p> <ul style="list-style-type: none">■ Machine: Displays the CPU usage of the machine.<ul style="list-style-type: none">□ User: Displays the CPU time used by user space processes.□ System: Displays the CPU time used by the kernel.□ lowait: Displays the time spent waiting for input or output operations.□ Idle: Displays the remaining CPU time that is not used actively.■ Self: Displays the CPU usage of the Virtuoso process.<ul style="list-style-type: none">□ CPU: Displays the CPU time used by the Virtuoso process.□ Threads: Displays the number of threads used by the Virtuoso process.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Memory</i>	<p>Provides a graphical representation based on time and usage (%) to calculate the memory performance. Displays the following parameters:</p> <ul style="list-style-type: none">■ Machine: Displays the total amount of memory available in the machine.□ Used: Displays the total amount of memory used.□ Free: Displays the amount of unused memory.□ Buff: Displays the amount of buffer memory.□ Cache: Displays the amount of cache memory.□ Avail: Displays the amount of memory available for starting new applications without swapping.□ Self: Displays the memory usage of the Virtuoso process.□ VIRT: Displays the total amount of virtual memory used by the Virtuoso process.□ RES: Displays the non-swapped physical memory that is used by the Virtuoso process.□ SHR: Displays the amount of shared memory available in the Virtuoso process.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Options

The following table describes the fields available on the *Options* tab of the Health Monitor Advanced form.

Field	Description
<i>Virtuoso Pulse</i>	Specifies the settings to monitor the Virtuoso Studio pulse measurements when enabled.
<i>Yellow Light Alert</i>	Specifies the threshold in seconds of consecutive high similarity call chains to report a concentrated pulse. Environment variable: <u>virtuosoPulseYellowLightAlert</u>
<i>Time to Start Auto Logging</i>	Specifies the threshold level in seconds of continuous Virtuoso Studio alerts or system event alarms to start an auto capture of logs. Environment variable: <u>autoLogLatency</u>
<i>Refresh Interval</i>	Specifies the interval in seconds to trigger a backtrace to get a callstack. Environment variable: <u>virtuosoPulseRefreshInterval</u>
<i>System Pulse</i>	Specifies the settings to monitor the System pulse measurements when enabled.
<i>Yellow Light Alert</i>	Specifies the threshold level in seconds of continuous system events to raise a system event alarm. Environment variable: <u>sysPulseYellowLightAlert</u>
<i>CPU Utilization</i>	Specifies the threshold level in percentage of high CPU utilization to report a concentrated pulse. Environment variable: <u>cpuUtilization</u>
<i>Memory Utilization</i>	Specifies the threshold level in percentage of high memory utilization to report a concentrated pulse. Environment variable: <u>memUtilization</u>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Refresh Interval</i>	Specifies the interval in seconds to measure the utilization of system resources. Environment variable: sysPulseRefreshInterval
Display	Specifies the display settings for the Health Monitor form when it is pinned.
<i>Toolbar</i>	Specifies the settings for the Health Monitor toolbar: <ul style="list-style-type: none">■ <i>Borderless</i>: Displays the Health Monitor without the form header. Environment variable: toolbarBorderless■ <i>Autohide</i>: Automatically hides the Health Monitor. The <i>Borderless</i> option must be selected for this option to be enabled. Environment variable: toolbarAutohide

Related Topics

[sysPulseRefreshInterval](#)

[virtuosoPulseRefreshInterval](#)

[Monitor by Strace Form](#)

[Health Monitor Overview](#)

[Health Monitor Form](#)

[Diagnostic Center Form](#)

[Collecting Data Using Health Monitor Tool](#)

Load Workspace Form

Use this form to choose a workspace to load for the current view type.

Field	Description
<i>Workspace</i>	Selects a workspace to load and apply to the current view type.

Related Topics

[Loading and Deleting a Custom Workspace](#)

[Setting the Default Workspace for an Application](#)

Make Read Only Form

Use this form to select one or more cellviews to make read-only. A prompt is displayed to save any modified data.

Field	Description
<i>All</i>	Allows you to make all the cellviews you have opened read only. If you have changed the data in any cellview, you get a prompt asking if you need to save the modified data.
<i>None</i>	Prevents any selected cellviews from being set to read only, retaining the edit state of the selected cellviews,
<i>For all modified Data</i>	Specifies an action for all the modified data. <ul style="list-style-type: none">■ <i>prompt</i>: Displays a prompt to lets you choose whether you want to save the changes.■ <i>save</i>: Saves the changes automatically.■ <i>discard</i>: Discards the changes automatically.
<i>Library</i>	Displays the name of the library whose cellviews you have opened for either viewing or editing.
<i>Cell</i>	Displays the name of all the cells corresponding to the library that you have opened for either viewing or editing.
<i>View</i>	Displays the names of the cellviews that you have opened.
<i>Window</i>	Displays the window number containing the cellview you had opened for either viewing or editing.
<i>Modified</i>	Indicates whether a cellview has been modified.

Related Topics

[Making Cellviews Read-Only](#)

[Getting a List of Locked Cellviews](#)

Monitor by Strace Form

Use this form to trace system calls of the specified process ID by specifying a particular data analysis mode.

Field	Description
<i>Strace Mode</i>	Specifies mode to perform data analysis of the system calls. <ul style="list-style-type: none">■ Normal: Provides data analysis including the runtime summary of the total number of system calls and top 10 system calls with the highest time duration.■ Summary: Provides a high-level summary of the system calls by specifying call counts and time duration.■ File status analysis: Provides statistics on the basis of the amount of time duration spent or the number of times a file is accessed.
<i>Start</i>	Starts gathering data to generate a report.
<i>Stop</i>	Stops gathering data and displays a report for the specified process ID.

Related Topics

[Health Monitor Form](#)

New Hierarchy Form

Use this form to create a new hierarchy.

Field	Description
Template	Displays available template and specifies name for it.
<i>Built-in</i>	Shows available templates for view lists, stop lists, and constants that contain more detailed hierarchy bindings. The default is <i>Other</i> ; it allows you to specify a new template.
<i>Name</i>	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 <i>Switch List</i> and <i>Stop List</i> fields.
Top Cell	Specifies the top cell of the configuration.
<i>Library</i>	Specifies the name of the library where the cell at the highest level (the root cell) of the configuration resides.
<i>Cell</i>	Specifies the cell at the highest level of the configuration (the root cell).
<i>View</i>	Specifies the name of the configuration file.
<i>Browse</i>	Opens the Library Browser.
Global Bindings	Lists libraries and cellviews for the configuration.
<i>Library List</i>	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.
<i>View List</i>	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.
<i>Stop List</i>	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.
<i>Description</i>	Enters a brief description of the configuration.

Related Topics

[Library Creation](#)

New Library Form (CIW)

Use this form to create a new library.

Field	Description
Library	Specifies the new library name and path.
<i>Name</i>	Assigns a name to the new library.
<i>Directory (non-library directories)</i>	Selects the directory into which you put the new library.
<i>Compression Enabled</i>	Check box writes OpenAccess data to library in a compressed format.
Technology File	Specifies the technology file setup.
<i>Compile an ASCII technology file</i>	Assigns a copy of the sample technology file (shipped with the software) to the new library for you to edit.
<i>Reference existing technology libraries</i>	References existing technology libraries.
<i>Attach to an existing technology library</i>	Opens a form that lets you select a technology file to attach to an existing library in your library definitions file.
<i>Do not need process information</i>	Creates the library without an attached technology file.
Design Manager	Selects the design management setup.
	If more than one design management option is available, a cyclic field is displayed with your choices, otherwise a single option gets displayed. When no design management environment is available, <i>No DM</i> gets displayed. In this scenario, your library opens with only basic design management features available.

Related Topics

[Library Creation](#)

Open File Form

Use this form to open a cellview and to specify the application in which you want it to get displayed.

Field	Description
File	Specifies the information related to the cell.
<i>Library</i>	Selects 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.
<i>Cell</i>	Specifies the name of the cell to open.
<i>View</i>	Specifies the name of the view to open.
<i>Type</i>	Displays the cell type.
<i>Browse</i>	Opens the Library Browser.
Application	Specifies the application to open the file.
<i>Open with</i>	Selects the application in which you want to open the file.
<i>Always use this application for this type of file</i>	Sets the specified application as default to open the specified file.
Open for	Specifies the use of the opened cellview.
<i>edit</i>	Displays the cellview and permits editing.
<i>read</i>	Displays the cellview.
<i>Library path file</i>	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.
<i>Cells</i>	Displays the names of all the cells in the library shown in the <i>Library Name</i> drop-down combo box.

Related Topics

[Opening a Cellview from the CIW](#)





Save .cdsenv file Form

Use this form to specify how modified environment variables are saved.

Field	Description
<i>Save to filename</i>	Specifies the <code>.cdsenv</code> file to which you want to save the changes.
<i>Saved tools</i>	<p>Enables you to specify the tools for which you want to save the environment variables. You can select:</p> <ul style="list-style-type: none">■ <i>all possible</i> Saves environment variable updates for all possible tools, includes both loaded and unloaded tools.■ <i>all loaded</i> Saves environment variables updates for tools that have been loaded.■ <i>loaded by name</i> Saves environment variables updates for only those tools that have been selected in the <i>Loaded Tools</i> tree.
<i>Saved variable states</i>	<p>Specifies the variables you want to save.</p> <ul style="list-style-type: none">■ <i>modified</i> Saves only those variables whose values have been edited.■ <i>modified and different from default</i> Saves updates to only those environment variables that have been modified and their current value is different from the default value.■ <i>all</i> Saves all environment variables in the selected tools.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Saved file status</i>	<p>Specifies the action taken when an existing <code>.cdsenv</code> file is updated:</p> <ul style="list-style-type: none">■ <i>overwrite</i> Overwrites the existing file.■ <i>merge (update file with most recent values)</i> Updates are merged in the existing file. Environment variables in the file are updated with the most recent value in the Cdsenv Editor.■ <i>retain (update file with additional information - retain file data)</i> Updates are added to the existing file, however, environment variables in the file retain their values.
<i>Loaded Tools</i>	<p>Displays the tree of tools that have been loaded. The tree becomes available when the <i>loaded by name</i> option has been specified under <i>Saved tools</i>. You can select the tools you want to save from this tree.</p>
	Expands all the supported tools in the Loaded Tools tree.
	Collapses all the supported tools in the <i>Loaded Tools</i> tree.
	Selects all the supported tools.
	Deselects all the supported tools.
<i>Preview</i>	<p>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.</p>

Related Topics

[Opening Cdsenv Editor in CIW](#)

Save As Form

Use this form to save a file.

Field	Description
<i>Look in</i>	Displays the directory to which you want to save the text file.
<i>File Name</i>	Displays 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.
<i>Files of type</i>	Displays the available file types for filtering what you see in the list area of the form. The default selection is <i>All Files (*)</i> .
<i>Save</i>	Saves the specified file and closes the form.

Save Cellviews Form

Use this form to save the latest changes made in the currently open cellviews.

Field	Description
<i>Save these cellViews before closing</i>	<p>Lists the library name, cell name, and view name of each cellview to which you have made unsaved changes.</p> <p>When the button to the right of the view name is selected, changes to that cellview are saved when you click <i>OK</i>.</p> <p>When the button to the right of the view name is not selected, changes to that cellview are not saved when you click <i>OK</i>.</p>
<i>All</i>	Indicates that all of the changed cellviews get saved when you click <i>OK</i> .
<i>None</i>	Indicates that none of the changed cellviews gets saved when you click <i>OK</i> .

Related Topics

[dbUndoAcrossSave](#)

[Close Opened Cellviews Form](#)

Save Defaults Form

Use this form to save default values for a tool.

Field	Description
Tools To Save	Specifies the applications, such as schematic, for which you want to save environment variables.
<i>All possible tools</i>	Specifies all applications in the <i>Available Tools</i> list box.
<i>All loaded tools (changed values only)</i>	Specifies all applications shown in the <i>Currently Loaded</i> list box. This option saves only the values that are different from default values.
<i>Loaded tools by name (changed values only)</i>	Specifies the applications you type in the <i>Tool Names</i> field. When you choose this option, the <i>Tool Names</i> field becomes editable. This option saves only the values that are different from default values.
Variables To Save	Specifies the criteria to use when saving environment variables. These options are available when you select the <i>All loaded tools</i> or <i>Loaded tools by name</i> options from <i>Tools to Save</i> .
<i>Modified variables only</i>	Specifies that only variables that have been modified in the session gets written out.
<i>Modified and different from Default</i>	Specifies that variables that have been modified with a value different from the default value gets written out.
<i>All tool variables</i>	Specifies all variables gets written out, whether modified or not.
<i>Available Tools</i>	Shows the applications for which you have a license.
<i>Currently Loaded</i>	Shows the applications for which environment variables are loaded.
<i>Tool Names</i>	Specifies the names of individual applications for which you want to save the default values. The default is ALL.
<i>Save To File</i>	Displays 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	Decides how to save the latest changes.
<i>Overwrite</i>	Overwrites your .cdsenv file.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Merge values</i>	Saves the values into your <code>.cdsenv</code> file. It does not delete pre-existing unmodified values.
<i>Retain values</i>	Saves the values to a new file. You must type a different file name in the <i>Save To File</i> field.

Related Topics

[Options Menu in the CIW](#)

Save Modified Data Form

Use this form to save modified or updated data.

Field	Description
<i>Select one or more objects to save from virtual memory</i>	Selects or deselects all open and modified cellviews and data in the form.
<i>All</i>	Selects all the cellviews in all the listed libraries.
<i>None</i>	Deselects any selected cellviews.
<i>Library</i>	Displays the names of the libraries containing the modified cellviews.
<i>Cell/Library File</i>	Displays the names of all cells or files corresponding to the library shown in the library column.
<i>View/Cell File</i>	Displays the name of the modified view. For example, <code>schematic</code> and <code>symbol</code> .
<i>View Type</i>	Displays the real view type of a cellview. In a cellview, view name can be user defined, for example, view name can be <code>layout1</code> . However, view type is always same for a specific type of data, such as <code>maskLayout</code> and <code>schematic</code> .
<i>Mode</i>	Indicates whether the cellview is in edit mode. Only data in edit mode is listed here.
<i>Status</i>	Displays the status of data in a specific window. The status is always <code>modified</code> , however, if a cellview is open in a window, the window number is included, for example, <code>win4:modified</code> , <code>win5:modified</code> and so on.

Related Topics

[Saving Modified Data](#)

[ddsHiSaveData](#)

Save Session Form

Use this form to save session information in a file.

Field	Description
<i>File Name</i>	Displays the name of the file in which you want to save the information. The default file name is <code>cdsSession.save</code> . You can rename this file.

Related Topics

[Options Menu in the CIW](#)

[Saving and Restoring Window Positions in Virtuoso](#)

Save View Form

Use this form to save a view.

Field	Description
<i>Name</i>	Specifies a name for the view.

Related Topics

[Saving and Restoring Views](#)

Scan/Repair Hierarchy Form

Use this form 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.

Field	Description
<i>Library</i>	Specify the library name of the top-level cellview.
<i>Cell</i>	Specify the cell name of the top-level cellview.
<i>View</i>	Specify the view name of the top-level cellview.
<i>Log File</i>	Specifies the name of the log file generated by oaScan. The default is <code>cdsScanHier.log</code> .
<i>Always View Log File</i>	Displays the log file on the desktop after each run.
<i>Repair</i>	Automatically repairs any issues found in the design hierarchy.
<i>Always Views in Cell</i>	Scans all the views present in the current cell.
<i>Verbose</i>	Prints log file information on all the designs processed rather than only those that have issues.

Related Topic

[Scanning a Hierarchy](#)

Scan/Repair Library Form

Use this form to scan an entire library or a specific design database for OpenAccess issues.

Field	Description
<i>Library</i>	Specifies the library to be scanned.
<i>Cell</i>	Specifies the name of a cell to be processed. If not specified, all the cells in the library are processed.
<i>View</i>	Specifies a view name to be processed. If the <i>Cell</i> option is specified, oaScan processes the cellview in question. If the <i>Cell</i> option is not specified, all views with the specified name are processed.
<i>ViewType</i>	Limits the scan to databases of the selected type.
Mode	Specifies the mode in which the design is processed.
<i>Scan</i>	Reports issues found but does not repair them.
<i>Repair</i>	Repairs any issues found in the database automatically.
<i>Checkout data when managed</i>	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 <code>oaScan.log</code> .
<i>Hide warnings</i>	Restricts the display of warning messages.
<i>Always View Log File</i>	Displays the log file on the desktop after each oaScan run.
<i>Verbose Output</i>	Lists log file information on all the designs processed rather than only those that have issues.

Related Topic

[Scanning a Library](#)

Scan On Save Form

Use this form to run oaScan automatically any time you save data to disk.

Field	Description
<i>Enable Scan On Save</i>	Switches on automatic checking during cellview save.
<i>Quiet (no dialog, auto-repair)</i>	Suppresses the pop up results dialog and attempts to automatically repair any issue found.

Related Topics

[Enabling the oaScan Utilities and Specifying the oaScan Version to Use](#)

Search Form

Use this form to search for a file using a specific text string.

Field	Description
<i>Search For</i>	Specifies the search string.
<i>When Found</i>	Specifies the action to be taken when the specified text is found.
<i>select</i>	Selects the found text.
<i>deselect</i>	Deselects the found text.
<i>scroll to next match</i>	Keeps scrolling until the next match is found.
<i>Match Options</i>	Specifies the match option for the specified text string.
<i>whole word</i>	Matches whole words in the text.
<i>exact match</i>	Matches those strings where the casing is the same as the specified search string.
<i>Wrap Around</i>	Searches wrap around text.
<i>Scan the Whole File</i>	Highlights all occurrence of the search string in the whole file.

Related Topics

[Searching a File for Specific Text](#)

Set Fonts Form

Use this form to view and edit the current font type settings.

Field	Description
<i>Label</i>	Specifies font type for the form labels.
<i>Text</i>	Specifies font type for the field text of forms.
<i>CIW</i>	Specifies the font type and size for the CIW.
<i>Use Text Font</i>	Specifies to keep the font settings same as <i>Text</i> .

Related Topics

[Viewing the Font List and Setting Fonts](#)

[hiSetFont](#)

Set Log File Display Filter Form

Use this form to select the outputs you want to display in the CIW.

Field	Description
Show	Indicates the items that gets displayed in the output area of the Command Interpreter Window.
<i>Return values from typed-in commands</i>	Displays return values from typed-in commands in the output area.
<i>Error message output</i>	Displays error message output in the output area.
<i>Warning message output</i>	Displays warning message output in the output area.
<i>Standard message output</i>	Displays standard message output in the output area.
<i>Accelerated input</i>	Displays accelerated input in the output area.
<i>Return values from accelerated input</i>	Displays return values from accelerated input in the output area.
<i>Prompt output</i>	Displays prompt output in the output area.

Related Topics

[Determining SKILL Function for a Menu Command](#)

Show File Form

Use this form to check the name of the file you wish to view.

Field	Description
<i>File Name</i>	Displays the name of the file to view. If you do not specify a path, the SKILL search path is used.

Related Topics

[Saving and Restoring Views](#)

SKILL Name Checker

Use this form to check for name collisions in the loaded contexts.

Field	Description
<i>Select contexts for checking name conflicts</i>	Specifies a valid context file to check the name conflicts. <ul style="list-style-type: none">■ <i>Add</i>: Moves the specified file name from the browser field to the list area.■ <i>Delete</i>: Removes the selected file from the list area.

Related Topics

[Conversion Toolbox Form](#)

[Surveyor: SKILL Tabulator Form](#)

Surveyor: SKILL Tabulator Form

Use this form to create a non-proprietary snap-shot of your SKILL environment and to report the Cadence functions called from your SKILL code. This form ignores all user functions to preserve confidentiality.

Field	Description
<i>File or Directory Name(s)</i>	<p>Specifies multiple space-separated SKILL file names or directory names containing the SKILL files to be tabulated.</p> <p>By default, the current directory from which you started the program appears in this field. This is a required field.</p>
<i>Do Not Resolve Symbolic Links</i>	<p>Enables the SKILL Tabulator to check the files that have no extension.</p> <p>By default, this check box is not marked: The SKILL Tabulator resolves symbolic links to check the files to which the links refer. For example:</p> <pre>% ls -l /usr/src/fileLn.il /usr/src/fileLn.il -> actualFile</pre> <p>When the check box is not marked, the SKILL Tabulator checks /usr/src/actualFile.</p> <p>You can mark this check box so that the SKILL Tabulator checks links without resolving them. In the example above, the SKILL Tabulator checks /usr/src/fileLn.il.</p>
<i>File Extensions</i>	<p>Specifies multiple extensions of files to be tabulated. By default, il ile cdsinit. If you use extensions that differ from the usual il ile cdsinit, you must edit this field. Unless you mark the <i>Do Not Resolve Symbolic Links</i> check box (above), the SKILL Tabulator follows symbolic links and the extensions are checked against the file to which the links refer. This is a required field.</p>
<i>Recurse Directories</i>	<p>Enables the SKILL Tabulator to recursively search all the files hierarchically under the directory path(s) you typed in the <i>File or Directory Name(s)</i> field. The default value is t.</p>
<i>Exclude Files or Directories</i>	<p>Specifies files or directories to be excluded from tabulation in the current directory level of the directories specified in the <i>File or Directory Name(s)</i> field.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Recursively Excluded Files</i>	Specifies the files to be recursively excluded from tabulation in the current directory level as well as all the subdirectories of the directories you typed in the <i>File or Directory Name(s)</i> field.
<i>Report File</i>	Specifies the output file (by default, <code>./skillTab.out</code>) that contains a report of Cadence and user-defined functions defined, called, and how many times they were called. This is a required field.
<i>Create Info File</i>	Enables the <i>Info File</i> field.
<i>Info File</i>	Specifies the name and location of an information file that the SKILL Tabulator creates containing a list of files where Cadence and user-defined functions were called.
<i>Create Defn File</i>	Enables the <i>Defn File</i> field.
<i>Defn File</i>	Specifies a definition file that the SKILL Tabulator creates containing a list of files where Cadence and user-defined functions were defined.
<i>Show Report</i>	Opens the report file in a viewing window when the tabulation is done.
<i>Show User Defined</i>	Lists the functions that are filtered by the local definition criterion in the output file.
<i>Customer Telephone Number</i>	Specifies your contact number, including area code or country code as required.
<i>Project Name</i>	Specifies the name of the project.
<i>Project Info</i>	Specifies the information related to the specified project.

Related Topics

[Operating SKILL Surveyor](#)

[Email Tabulated List of Functions Form](#)

Software Product License Management Form

This form displays all the licensing details.

Field	Description
<i>Checked Out Licenses</i>	Displays the licenses that are currently running on your desktop. This pane updates when licenses are checked out by the applications or are started in the <i>Task Assistant</i> menu in the editing environments.
<i>Relevant Licenses</i>	Displays the licenses that are available with the platform that is running.
<i>Token Information</i>	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.
<i>Users</i>	Displays token license usage and availability of the license you highlight. The <code>Control</code> space deselects highlighted licenses.

Related Topics

[Starting Virtuoso Studio](#)

[Configuring the Virtuoso Studio](#)

Unable to check out Form

When all available licenses have been checked out, this form lists the licenses that are currently in use, and also displays the list of active users.

Field	Description
<i>Product</i>	Identifies the product number that cannot be checked out.
<i>Licenses</i>	Specifies how many licenses are available for the application.
<i>User</i>	Identifies current users who have a license checked out for the application.
<i>Host</i>	Specifies the host system where the application resides.
<i>Product Description</i>	Provides the name or short description of the application.

Related Topics

[Auto Checkout Preferences Form](#)

User Preferences Form

Use this form to specify user preferences for the current session.

Field	Description
Window Controls	Specifies window controls, which are settings that affect the position and appearance of windows in the Virtuoso user interface.
<i>Place Manually</i>	<p>Controls the window placement.</p> <ul style="list-style-type: none">■ 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, it determines 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.
<i>Create New Window When Descending</i>	<p>Controls the type of view displayed when you go up and down the design hierarchy. Not all applications make use of this feature.</p> <ul style="list-style-type: none">■ When turned off, continues to display the original view type.■ When turned on, displays another window with the new view type.
<i>Scroll Bars</i>	<p>Controls the visibility of scroll bars.</p> <ul style="list-style-type: none">■ 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.
<i>Tear-Off Menus</i>	<p>Controls whether future menus have tear-off handles.</p> <ul style="list-style-type: none">■ When turned off, future menus have tear-off handles.■ When turned on, future menus do not have tear-off handles.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Menu Shortcuts</i>	<p>Controls whether menu shortcuts are available.</p> <ul style="list-style-type: none">■ When checked, you can type <i>Alt</i> together with the underlined character for a menu (such as the <i>E</i> for the <i>File</i> menu in the CIW) to access that menu, and you can type the underlined character on a menu item (such as the <i>O</i> in <i>Open</i> on the <i>File</i> menu) to access that item.■ When not checked, you cannot access menus or menu items using keyboard shortcuts.
<i>Mouse Prompts</i>	<p>Controls whether and where mouse prompts appear.</p> <ul style="list-style-type: none">■ <i>Top</i> if you want mouse prompts to appear along the top edge of each new window.■ <i>Bottom</i> if you want mouse prompts to appear along the bottom edge of each new window.■ <i>None</i> if you do not want the mouse prompts to appear at all. <p>The <i>Mouse Prompts</i> preferences are overridden by any applied workspace regardless of the selected position. Also, if the mouse prompts are set to <i>None</i> 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.</p>
<i>Focus Policy</i>	<p>Selects the focus policy if the pointer remains in a session window canvas for a specific period of time.</p> <ul style="list-style-type: none">■ <i>Click</i> to click in the session window or the dockable window to be in focus.■ <i>CanvasDelay</i> 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.■ <i>CanvasAsstDelay</i> 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. <p>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.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Focus Delay</i>	<p>Sets the specific period of time for which the pointer must remain in a session window canvas to get focus.</p> <p>The default value is 200 milliseconds, which is 1/5 of a second.</p>
<i>Default Editor Background Color</i>	<p>Changes the background color of the editor window. On clicking this option, the Select Color window displays, from where you can choose the required color.</p> <p>Default: <code>Black</code></p>
<i>Side Dock Tabs</i>	<p>Selects the display of the current tabs.</p> <ul style="list-style-type: none"> ■ <i>Bottom</i> if you want the tabs for multiple assistants, which are open at the same time, to appear at the bottom of a session window. ■ <i>Side</i> if you want the tabs for multiple assistants, which are open at the same time, to appear on the side,
Command Controls	Controls the Option forms and the mouse.
<i>Infix (No Click is necessary for first point)</i>	<p>Controls mouse behavior in design windows.</p> <ul style="list-style-type: none"> ■ 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. <p><i>Infix and Options Displayed When Commands Start</i> 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.</p>
<i>Options Displayed When Commands Start</i>	<p>Controls the behavior of options forms.</p> <ul style="list-style-type: none"> ■ When turned off, does not display an options form for changing command settings. Press <code>F3</code> 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.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Undo Limit</i>	Specifies how many commands you can undo with the <i>Undo</i> command. <i>Undo</i> 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.
<i>Nest Limit</i>	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.
<i>Double Click Time</i>	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.
<i>Beep Volume</i>	<p>Sets the volume of the beep used in the Cadence software. You can select intervals from -100 to +100 by dragging the slider.</p> <p>For systems that do not support volume control, any setting from -99 to 100 turns on the beep sound.</p>
<i>Web Browser</i>	Specifies a browser executable. The directory where the executable is must be in your path. The default is <code>firefox</code> .
CIW Controls	Controls the appearance and configuration of the Command Interpreter Window.
<i>Output History</i>	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.
<i>Input Buffer Lines</i>	Specifies the number of previous commands that appear in the input area.
<i>Input Area Lines</i>	<p>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.</p> <p>Additionally, the height of the CIW also get 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.</p> <p>Environment variable: <u><code>ciwCmdInputLines</code></u></p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Output Wrap Mode</i>	<p>Wraps the messages in the CIW.</p> <ul style="list-style-type: none">■ <i>None</i> if you do not want to wrap any messages that appear in the CIW. This is also the default value.■ <i>Word</i> if you want to wrap a message between words. This means that words longer than the width of CIW is going to appear in the next line and a horizontal scroll bar is displayed■ <i>Anywhere</i> if you want to wrap a message between characters. This means that words longer than the width of CIW is going to appear broken and a horizontal scroll bar get displayed■ <i>Word or Anywhere</i> if you prefer wrapping between words but in case a word is wider than the CIW, then the messages are going to get wrapped between characters. Therefore, in this case a horizontal scroll bar get displayed.
<i>Retain Unique Command</i>	<p>Removes identical commands from the input area of the CIW. By default, it is set to <code>nil</code>.</p>
<i>Output Area Lines</i>	<p>Specifies the maximum number of lines contained in the output area.</p> <p>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.</p> <p>The minimum valid value for this setting is 100.</p>
<i>Syntax Highlighting</i>	<p>Enables checking the syntax or commands entered in the input area of the CIW. By default, this check box appears selected.</p> <p>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.</p> <p>For more information, see Running SKILL Commands.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>History In Place</i>	<p>Accesses 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 navigation keys. If the check box is not selected, the up arrow key traverses 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.</p> <p>The CIW built-in history is different from using the <code>history()</code> command. The built-in history only displays typed-in commands, whereas, the <code>history()</code> command also lists accelerated commands, those commands that were run by selecting menus or using bindkeys.</p> <p>For information about the history command, see history().</p>
<i>Tab Stop</i>	<p>Specifies the number of character spaces that is going to represent a tab in both, the input and output areas of the CIW.</p>
<i>Syntax Highlighting</i>	<p>Uses any of the following key combinations: <code>Shift+Return</code>, <code>Shift+Enter</code>, <code>Ctrl+J</code>, or <code>Ctrl+M</code> to enter a newline when the <i>Enter Key Executes Command</i> check box appears selected.</p> <p>When the <i>Syntax Highlighting</i> check box is not selected, all of the key combinations execute the command block, that is, send the command block to the SKILL parser. This is because if the commands are not complete, they are not going to be executed by the SKILL parser until SKILL receives the remainder of the commands.</p>

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Enter Key Executes Command</i>	<p>Specifies whether or not you can press <i>Enter</i> to execute the command typed on the input line.</p> <p>When turned on, you can press <i>Enter</i> to execute the command typed on the input line.</p> <p>The command block that contains the cursor gets executed regardless of the position of the cursor within that block. In the following example, ^ marks the position of the cursor when you press <i>Enter</i>:</p> <pre>foreach(term dbGetNetParent printf("%s\n" term->name())) ^</pre> <p>The program evaluates and executes the entire command block.</p> <p>When turned off, you must place the cursor at the end of the input line and press <i>Enter</i> to execute the command.</p> <p>The settings of the <i>Enter Key Executes Command</i> only apply if the <i>Syntax Highlighting</i> check box is selected. For more information, see Running SKILL Commands.</p>
<i>Raise CIW on</i>	<p>Specifies whether the CIW should be raised to the front when a <i>Warning</i> and an <i>Error</i> has been displayed in the log window. The default is to not display the CIW in either of these situations.</p> <p>Environment variables: raiseCIWonError and raiseCIWonWarning</p>
Dashboard Controls	Controls the display of license activity indicators.
<i>Display Dashboard Indicators</i>	Specifies whether or not to display dashboard indicator information, related to resource license activity.
<i>License Activity</i>	Specifies where dashboard controls are displayed, either in the <i>CIW</i> , the <i>CIW+Session</i> windows, or that they are <i>Hidden</i> .

Related Topics

[File Preferences Form](#)

ViewFile Window Form

Views 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.

Field	Description
File	Specifies an action for the viewfile window.
<i>Open</i>	Opens a file for viewing in the current viewfile window
<i>Save</i>	Saves the information in the viewfile window to the log file
<i>Save As</i>	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.
<i>Auto Update On</i>	Causes the simulator to update the information in the view window automatically as the simulation continues.
<i>Auto Update Off</i>	Prevents the simulator from updating the information in the viewfile window as the simulation continues.
<i>Close Window</i>	Closes the viewfile window.
Edit	Edits the viewfile window.
<i>Copy</i>	Copies the selected text to the clipboard. You can also click the <i>Copy</i> command before selecting the text to be copied.
<i>Select All</i>	Selects the entire contents of the file

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Find</i>	<p>Searches for text strings in the viewfile window.</p> <p><i>Search for</i> specifies the text string you want to find.</p> <p><i>When Found</i> specifies whether to select or deselect the text string.</p> <ul style="list-style-type: none">■ <i>select</i> highlights the text string.■ <i>deselect</i> turns off highlighting of the text string.■ <i>scroll to next match</i> 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. <p><i>Match Options</i> control for partial hits and case.</p> <ul style="list-style-type: none">■ <i>whole word</i> specifies a search for the exact string.■ <i>exact case</i> specifies a search for the case (upper or lower) that appears in the <i>Search for</i> field.■ <i>Wrap Around</i> (default) continues the search from the beginning of the file.■ <i>Scan the Whole File</i>, when enabled, finds and highlights all occurrences of the text string you specify.
<i>Go to Line</i>	<p>Searches for the line number you specify and highlights the specified line.</p> <p><i>Go to line</i> field specifies the line number to which you want to move to.</p>
View	Specifies the display option for the viewfile window.

Virtuoso Studio Design Environment User Guide

Virtuoso Studio Design Environment Forms

Field	Description
<i>Line Numbers</i>	<p>Displays the following sub-menu options:</p> <ul style="list-style-type: none">■ <i>Inline</i>: Shows the line numbers on the left side (sidebar) of the log file. <p>You can enable sidebar by using the following variable in CIW:</p> <pre>(envSetVal "ui.text" "showLineNumbersInSideBar" 'boolean t)</pre> <p>Otherwise, to enable sidebar, you can add the following syntax in the <code>.cdsenv</code> file:</p> <pre>ui.text showLineNumbersInSideBar boolean t</pre> <ul style="list-style-type: none">■ <i>Status Bar</i>: Displays the line and column number in the lower-right corner of the log file.
<i>Highlight Current Line</i>	Highlights the current line in gray.

Related Topics

[Functions of Viewfile Window](#)

[Viewing the Hierarchy Tree](#)

What's New Search Form

Use this form to search for a particular document or feature.

Field	Description
<i>Search For</i>	Specifies a search string.
<i>Match Entire String</i>	Specifies whether you want the program to match the entire string.
<i>yes</i>	Indicates that you want the program to match the entire search string that you typed in the <i>Search For</i> field.
<i>no</i>	Indicates that you want the program to match words within the search string.
<i>Match Case</i>	Specifies whether you want the program to match casing - uppercase or lowercase.
<i>yes</i>	Indicates that you want the program to match casing.
<i>no</i>	Indicates that you want the program to ignore casing when finding a match.
<i>Wrap Search</i>	Specifies whether the program should wrap around to the top of the search document when it comes to the end.
<i>yes</i>	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.
<i>no</i>	Indicates that you want the program to stop searching when it comes to the end of the search document.

Related Topics

[Get Help with Virtuoso Menus](#)

Virtuoso SKILL Interface

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 to communicate within and between applications and designs. You can use SKILL for small tasks, such as generating a single command, or to build complex routines to customize your system and add functionality.

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 SKILL functions supported by the Virtuoso Studio Design Environment.

To access the SKILL Finder:

- ➔ From the CIW, choose *Tools – SKILL Finder*.

The Cadence SKILL API Finder window opens.

You can enter SKILL commands from the input line of the CIW or place them in command files. The SKILL development tool 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.

Running 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.

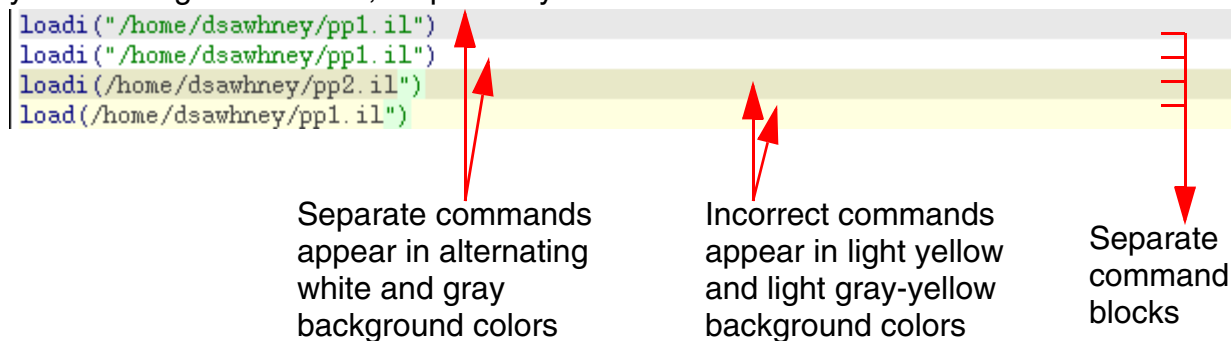
To insert a new line / command, press any of the following key combinations in the edit area: `Shift+Return`, `Shift+Enter`, `Ctrl+J` or `Ctrl+M`.

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.

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.

SKILL Commands Execution

The 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.

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.

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 only gets copied to the edit area as text.

For a detailed list of bindkey combinations that you can use to select and execute commands, see "[Bindkeys and Access Keys](#)".

Debug Support for SKILL Commands

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

Virtuoso Studio Design Environment User Guide

Virtuoso SKILL Interface

`envSetVal()` function or specifying them in your `.cdsenv` file at any time. The various syntax elements are as follows:

Syntax	Description
<code>ciwCommentColor</code>	Used for single line and multi-line comments Single line comments begin with a semicolon (;) and multi-line comments are surrounded by <code>/*</code> and <code>*/</code>
<code>ciwParenColor</code>	Used for parentheses ()
<code>ciwBracketColor</code>	Used for square brackets ([])
<code>ciwBraceColor</code>	Used for curly braces ({ })
<code>ciwSymbolColor</code>	Used for explicit symbols and for numeric ranges, such as <code>"0:10"</code> or <code>"3.4:4.8"</code> Explicit symbols are words preceded by a single quote (')
<code>ciwKeywordColor</code>	Used for the known set of SKILL keywords, such as <code>abs</code> , <code>acos</code> , <code>cdar</code> , <code>cddr</code> , and so on
<code>ciwStringColor</code>	Used for strings enclosed within double quotes
<code>ciwMatchParenColor</code>	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.

Syntax	Description
<code>ciwMismatchParenColor</code>	<p>Used as the background color to highlight mismatched parentheses, braces and brackets.</p> <p>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.</p> <p>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.</p>
<code>ciwUnmatchQuoteColor</code>	<p>Used as the background color for strings that do not have a closing double quote, or that start and end on different lines</p>
<code>ciwMismatchCmdColor1</code>	<p>Used to indicate an incomplete syntax status for the command blocks that have a white background</p> <p>Syntax is said to be incomplete if it contains unmatched parentheses, braces, brackets, double quotes or multi-line comments.</p> <p>Note: Syntax complete commands are not necessarily syntax correct.</p>
<code>ciwMismatchCmdColor2</code>	<p>Same as <code>ciwMismatchCmdColor1</code> only that this is used for command blocks that have a light gray background</p>

Command Repetition in 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 executes 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 searches for commands that contain “`iFo`”, such as `hiFormCancel`.

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

```
!!
```

- 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 edit area to display more 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.

Related Topics

[User Preferences Form](#)

[Tools Menu in the CIW](#)

Setting the SKILL Search Path and Specifying the Editor

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:

- ➔ 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:

- ➔ 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`).

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:

- ➔ Add the following line to your `.cdsinit` file:

```
editor = "xterm -e emacs"
```

Related Topics

[Operating SKILL Surveyor](#)

Operating SKILL Surveyor

The SKILL Surveyor provides access to the SKILL Tabulator from the *CIW – Tools – Conversion Tool Box*. To run the SKILL Surveyor, do the following:

1. In the Conversion Tool Box, click *SKILL Surveyor*.

The Surveyor: SKILL Tabulator form appears.

An asterisk (*) in front of a field label indicates a required field. You must type data in these fields. Accuracy is important because some of these fields are used to file PCRs.

2. Fill out the fields on the Surveyor: SKILL Tabulator form according to the field descriptions outlined in SKILL Tabulator Form.
3. Click *OK* or *Apply*.

If the SKILL Surveyor can locate a valid mail command on your system (`mail` or `mailx` in `/usr/bin` or `/bin`), the Email Tabulated List of Functions form appears.

If the SKILL Surveyor is not able to locate a valid mail command on your system, the Specify Alternative Mail Command Path form appears.

4. Fill out the required form or forms and click *OK*.

The SKILL Tabulator creates a non-proprietary snap-shot of your SKILL environment and reports the Cadence functions called from your SKILL code. It ignores all user functions to preserve confidentiality.

The SKILL Tabulator generates three files:

- `skillTab.out` (list of functions defined and called and how many times called)
- `skillTab.info` (files where functions were called)
- `skillTab.defn` (files where functions were defined)

You can email `skillTab.out` to Cadence Support for analysis. Cadence Support emails the analysis results back to the e-mail address you type in the *Customer Email Address* field on the SKILL Tabulator form.

The analysis provides the following information:

- All undefined functions (misspelled or dependencies): You called a function that neither you nor Cadence defined.
- All re-defined Cadence functions: You defined a function that Cadence also has defined in the code.

- All changed functions (most are positive improvements): You called a function that has changed between releases.
- All deleted functions (with replacements if available): You called a function that has been deleted between releases.
- All functions that are safely supported: You called a function that we document and support.
- All functions that might be dangerous to use: You called a function that is not documented or not supported by Cadence.
- Advice on how to recover from any changes identified.

Specifying Alternative Mail Command Path

When you click *OK* on the SKILL Tabulator form, this form appears only if the SKILL Surveyor is not able to locate a valid mail command on your system (`mail` or `mailx` in `/usr/bin` or `/bin`).

To specify an alternative path to a mail command, do the following:

- ➔ In the *Enter an alternative mail command path* field, type an alternative path.

Troubleshooting SKILL Issues in Virtuoso

Here is a list of commonly occurring SKILL issues:

- Unable to mail the report file

A command failure has occurred. For example, the `ilmail` script might be missing in `cds_root>/tools/bin`. Look in your terminal window for the corresponding error message.
- SKILL tabulation report file not found. Possible causes are as follows:
 - ❑ You left the *Report File* field on the SKILL Tabulator form empty or you do not have write permission for the file path you specified.
 - ❑ The specified *Report File* file was accidentally removed. You can run the tabulation again to regenerate this file.
 - ❑ Bad SKILL files caused the tabulation run to fail. Refer to your log file (for example, `~/CDS.log`) for more information about the errors and make sure that all of your SKILL files have no syntax errors before running tabulation again.

Virtuoso Studio Design Environment User Guide

Virtuoso SKILL Interface

Related Topics

[Surveyor: SKILL Tabulator Form](#)

[Email Tabulated List of Functions Form](#)

Diagnostics

This topic gives you an overview of the functions related to diagnostic testing and analysis:

- **Application crash report:** A crash report gets generated whenever Virtuoso application experiences any failure or crash. You can access this crash report from the link displayed in the Fatal Application Error dialog box, which appears after an unexpected termination of an application.
- **Graphics Performance Measurement:** It evaluates the ability of a platform to run various graphics operations required by the Virtuoso applications.
- **Health Monitor:** When an application slows down or freezes, this tool helps in debugging the issues by creating a high-level summary and by collecting relevant data.

Related Topics

[Application Crash Reporting](#)

[Measuring Graphics Performance](#)

[Health Monitor Overview](#)

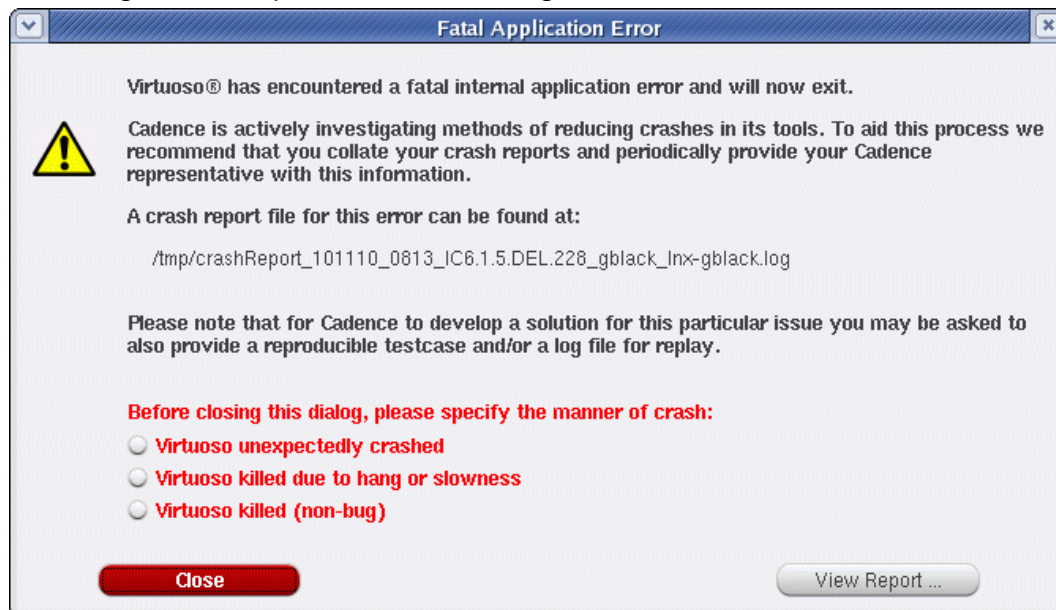
Application Crash Reporting

In Virtuoso, a report gets 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.

A crash report is not intended to provide a description of all possible debugging techniques, nor does it provides 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 by selecting *View Report* from the dialog box.



- Before closing the Fatal Application Error dialog, specify the nature of the Virtuoso crash, as this assists Cadence in its software crash investigations. In addition to the Crash Report, the option you select gets stored in `STARTINFO_FILE`.

- ☐ Select *Virtuoso unexpectedly crashed* if the software terminated unexpectedly without manual intervention.

If this option is selected, the following gets added to the Crash Report window:

```
signal:Abort(6) ****PROCESS CRASHED****
```


Virtuoso Studio Design Environment User Guide

Diagnostics

- ❑ 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 gets 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 gets 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 gets added to the Crash Report window:

```
signal:Abort(6)
```

- If a `SEGVFAULT` happens (again) inside the `virtuoso` crash hooks, the signal gets delivered directly to, and processed by, the crash report.
- If `virtuoso` is being run in `nograph` mode, the Fatal Application Error window does not get displayed. Rather, crash report information is 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 going to focus attention on genuine software crashes.

If you chose *View Report*, the Crash Report window is displayed.

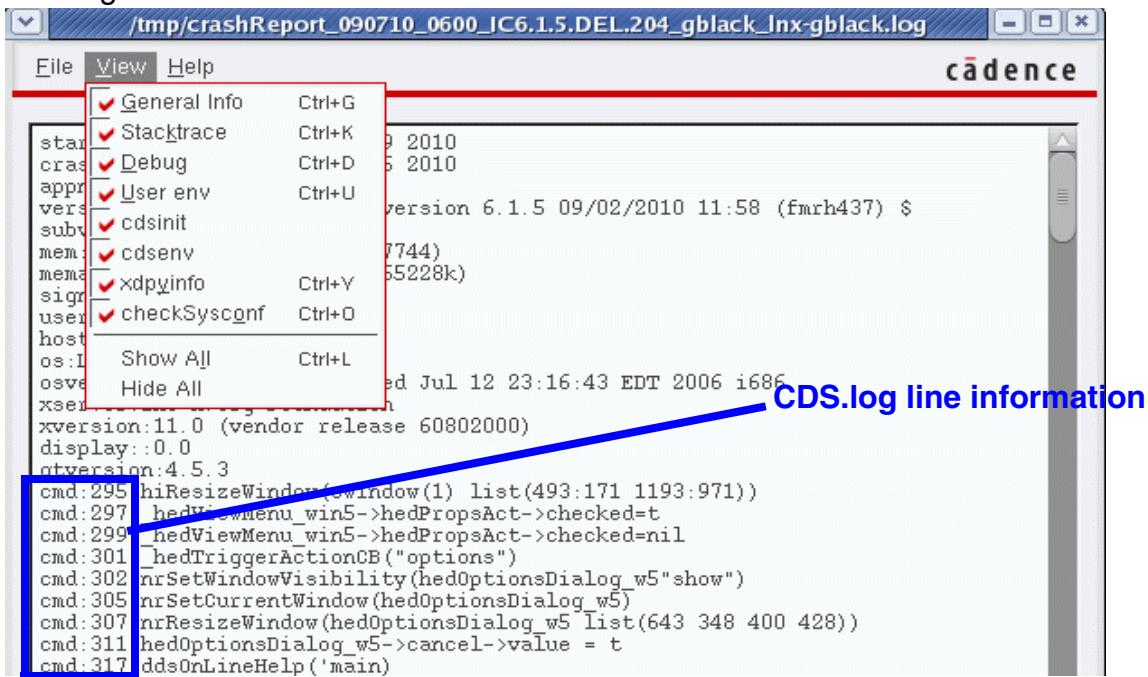
- From here, you can choose to save the contents of the report to a text file (*File – Save As*) for re-use, or select *File – Close* to exit the report after viewing.
- Additionally, within the *File* menu, you can choose to *Copy* report text, *Select All* report text, or *Find* particular report information 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.

On crash, a copy of the `CDS.log` file gets 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 additionally saves the date and time of crash. Also, in case Virtuoso crashed because of an external signal, the signal name also get 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.

Virtuoso Studio Design Environment User Guide

Diagnostics

- The crash report also prints any crash signals sent by a user
- 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.



Related Topics

[Crash Report Customization](#)

Crash Report Customization

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.

Name-Value Pair	Description
<code>EXTENDED=TRUE</code>	<p>If set to <code>TRUE/YES</code>, additional system information gets recorded for analysis (including the values for of <code>.cdsenv</code> variables, for all loaded tools, that differ from their default values).</p> <p>The default setting is <code>TRUE/YES</code>.</p>
<code>ALWAYS=NO</code>	<p>If set to <code>TRUE/YES</code>, the debug information always gets created and the Fatal Application Error dialog no longer required to be displayed.</p> <p>The default setting is <code>NO/FALSE</code>.</p>
<code>STARTINFO_FILE=abcd.out</code>	<p>Used to log each session start. Specify the name of the file to log each session start/end.</p>
<code>SCRIPT</code>	<p>Executes the shell/script file of a user when any application crashes. You can set the variable as:</p> <pre>setenv CDS_ERRORLOG "SCRIPT=<directory path>/ <file name>.sh"</pre> <p>The user script file passes the following arguments:</p> <pre>-status <exit_status> -log <path_to_CDS.log_file> -t <crash_time> -dir <log_directory> -exe <path_to_application_binary></pre>
<code>DIR=.</code>	<p>Used to set the directory that reports are to be saved in.</p> <p>If you have specified a directory that does not currently exist, this directory gets created automatically in this case.</p> <p>If this environment variable is not set, reports gets saved to <code>"/tmp"</code> (the name generated by <code>cdsGetTmpDir()</code>) by default.</p>

Virtuoso Studio Design Environment User Guide

Diagnostics

Name-Value Pair	Description
AUTO_DIR=<path>	<p>If this environment variable is set, crash reporter creates a directory <path> and all crash logs are stored in <path>/year_month.</p> <p>For example:</p> <pre>>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_IC6.1.8- 64b.500.8_user_msc075.cadence.com.log -rw-rw-rw- 1 user cadence10 4252 Dec 8 19:18 crashReport_120816_191828_IC6.1.8- 64b.500.8_user_msc075.cadence.com.log.CDS.log</pre> <p>If AUTO_DIR or DIR environment variables are not set, reports gets saved to "/tmp" (the name generated by cdsGetTmpDir()) by default.</p>
ALL_ERRORS=TRUE	<p>Generates crash reports.</p> <p>This value does not work for SIGTERM, SIGHUP, SIGALRM, and XIO errors that can cause Virtuoso to crash. Additional information regarding these signals is added to the CDS.log file before Virtuoso exits. This additional information describes the event that occurred, the state of the Virtuoso process, and the state of the host machine.</p> <p>It is recommended to use the Health Monitor tool to gather information about issues with a Virtuoso process, instead of sending signals to terminate the Virtuoso application, and then reporting it as a crash.</p>
NAME_FMT=%PID.log	<p>Used to define the report name.</p>

Virtuoso Studio Design Environment User Guide

Diagnostics

Name-Value Pair	Description
USE_DEBUGGER=YES	<p>Attempts to attach the platform debugger on application crash to get additional stack information.</p> <p>The default setting is TRUE/YES.</p> <p>If USE_DEBUGGER has been set to YES, but no debugger has been set in the path, then the crash report informs that a debugger was not available on exit.</p> <pre>debug:Warning : no platform debugger found debug:Check your system configuration</pre>
CDS_CRASH_DBG_SCRIPT	<p>You can specify a different third-party platform debugger using the CDS_CRASH_DBG_SCRIPT environment variable to get additional stack information when an application crashes.</p> <pre>setenv CDS_CRASH_DBG_SCRIPT "/opt/install/ bin/ my_debug.sh %PID %PROGRAM"</pre> <p>For example:</p> <pre># 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</pre> <p>In case the user debugger cannot start, the following message is added to the crash report:</p> <pre>CDS_CRASH_DBG_SCRIPT was not started</pre>

Virtuoso Studio Design Environment User Guide

Diagnostics

Name-Value Pair	Description
DISABLED=NO	<p>Disables the use of crash reporting so that no crash dialog is displayed nor a crash report generated.</p> <p>The default setting is <code>FALSE/NO</code>.</p>
SAVELOGS=ONCRASH	<p>Lets you specify that <code>virtuoso</code> calls the <code>saveCdsLog.pl</code> log collection script at any exit, or only on crash, to automatically save the current <code>CDS.log</code>.</p> <p>Accepted values are: <code>ALWAYS/ONCRASH/OFF</code>. The default is <code>ONCRASH</code> which saves a copy of <code>CDS.log</code> to the crash directory on exit.</p>
SIGTERM=TRUE	<p>Lets you terminate the Virtuoso application. This name-value pair allows Virtuoso to release resources and save its state, if appropriate.</p> <p>The default value is <code>FALSE</code>.</p>

Virtuoso Studio Design Environment User Guide

Diagnostics

Name-Value Pair	Description
LOG_EMAIL	<p>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.</p> <p>The format of the e-mail is as follows:</p> <p>To:</p> <p><application1>:<address1,address2>,<application2>:<address3,address4></p> <p>Subject: Virtuoso log info for host:<hostname> (lnx86)</p> <p>The content of the e-mail is as follows:</p> <p>INST_ROOT: <Installation root of the product></p> <p>VERSION:<Product version></p> <p>USER: <User name></p> <p>TIME: <Timestamp of the crash></p> <p>REPORT: <Path of the crash report></p> <p>LOG: <Path of the CDS.log file></p> <p>SIGNAL: <Signame# and its description></p> <p>If <code>\$CDS_ERRORLOG LOG_EMAIL=<E-Mail list></code> is set, then in case of application crash, the first 50 lines of <code>CDS.log</code> comes as an attachment in an e-mail.</p>

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"
```

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 gets 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`.

Related Topics

[Additional Crash Data Collection](#)

[Crash Trend Reporting](#)

[Crash Report Data Storage](#)

[Health Monitor Overview](#)

[Viewing the cds.lib Updates](#)

Crash Trend Reporting

You can track session start and end data using the name-value pair `STARTINFO_FILE` with the `CDS_ERRORLOG` environment variable.

Note: It is recommended that session logging information only be used on a local, rather than global, site basis to ensure that Virtuoso performance is not impacted.

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 gets logged. 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 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 gets 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` or `/tmp`.

For example:

```
[ 7555@mymachine@jsmith ] start date/time:Sat Mar 21 06:37:57 2009 virtuoso
IC6.1.4.355 plat:sun4v
[ 7555@mymachine@jsmith ] crash date/time:Sat Mar 21 06:38:26 2009 virtuoso
IC6.1.4.355 plat:sun4v
crash report file :: /tmp crashReport_032109_0638_IC6.1.4.355_jsmith_mymachine.log
```

`STARTINFO_FILE` contains the name of the `startinfo` in `CDS_ERRORLOG`, or in `/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.

If you shutdown a virtuoso process using the `kill -9` command, the `STARTINFO_FILE` does not record the exit/crash.

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` displays it as follows:

```
[7555@mymachine@jsmith] 11(SegFault):Wed Jul 8 13:16:03 2009 virtuoso IC6.1.4.404  
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 IC6.1.6-  
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 uses 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 2009 virtuoso  
IC6.1.4.375 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 returns `NULL`, and output that information into the X server section of the crash report. For example, (`xserver: (Not Responding)`).

If virtuoso is being run in `-nograph` mode this check does not get 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 gets displayed in the `STARTINFO_FILE`:

```
[7555@mymachine@jmisth] exit(111):Thu Jun 11 12:35:43 2009 virtuoso IC6.1.4.387  
sun4v
```

Related Topics

[Crash Report Customization](#)

[Crash Report Data Storage](#)

Additional Crash Data Collection

If you require additional crash data to be collected, you can set the `EXTENDED` name-value pair for the `CDS_ERRORLOG` environment variable to be `TRUE`.

If `EXTENDED` is activated, the crash report also records the details of your environment variable settings and the content of the `.cdsinit/cdsenv` files.

- Where for `.cdsinit`, the content of the `.cdsinit` file gets included in the crash data.
- Where for `.cdsenv`, the name/value of those `cds` environment variables that differ from their default values gets included in the crash data.

It also reports `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 gets reported.
- Where for `xdpyinfo`, the results of checks on the available graphics display visuals gets reported.

`EXTENDED` is set to `TRUE` by default.

Related Topics

[Crash Report Customization](#)

[Crash Report Data Storage](#)

Crash Report Data Storage

Crash reports are saved into the directory specified by the `DIR` name-value pair using the `CDS_ERRORLOG` environment variable. 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, then this directory gets created automatically.

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_IC6.1.4.302_jsmith_msc080.log
```

`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.

Related Topics

[Crash Report Customization](#)

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. It 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.

Examples of graphic performance measurements include on-screen redraw performance, area copy performance, and image operation performance.

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`

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 Results Window.

4. Optionally, run the *Calibrator* (graphics measurement test tool) by clicking the *Run* button, or by choosing *Tools – Calibrator*.

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.

Related Topics

[Benchmark Results Window Form](#)

[Graphics Performance Benchmarks Window Form](#)

hiGraphicsBenchmark Command-Line Arguments

The following command-line arguments can be used with `hiGraphicsBenchmark`:

- | | |
|----------------------------------|---|
| <code>[-taskviewer [PID]]</code> | Runs <code>hiGraphicsBenchmark</code> as a task viewer. Optionally, a process <code>PID</code> can be specified to monitor its CPU usage, memory usage and so on. If <code>PID</code> is not specified, it searches for any <code>virtuoso</code> process, or use the current process as default. |
| <code>[-calibrator]</code> | Runs <code>hiGraphicsBenchmark</code> as a calibrator to test key graphics operations, and provide alternative solutions, through settings or command line operations so as to improve graphics performance. |
| <code>[-h -H]</code> | Displays <code>hiGraphicsBenchmark</code> . |

Related Topics

[Graphics Performance Benchmarks Window Form](#)

[Performance Benchmarks](#)

[Task Viewer](#)

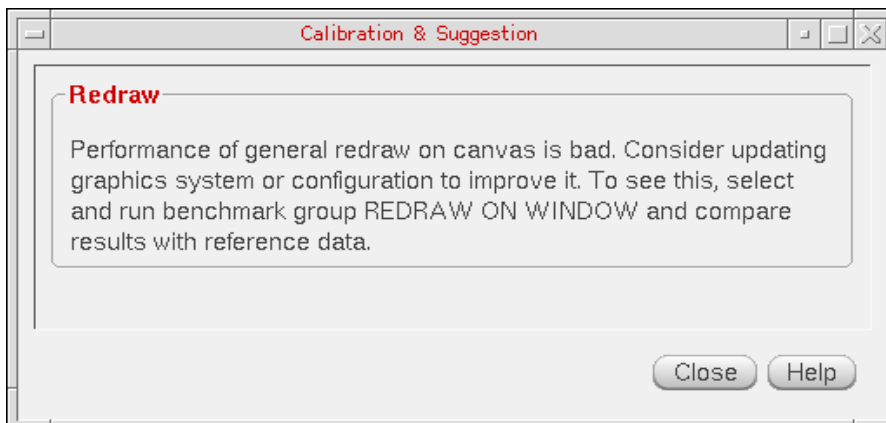
[Performing Benchmark Tests](#)

Performing Benchmark Tests

There are two methods that can be used to perform benchmark tests.

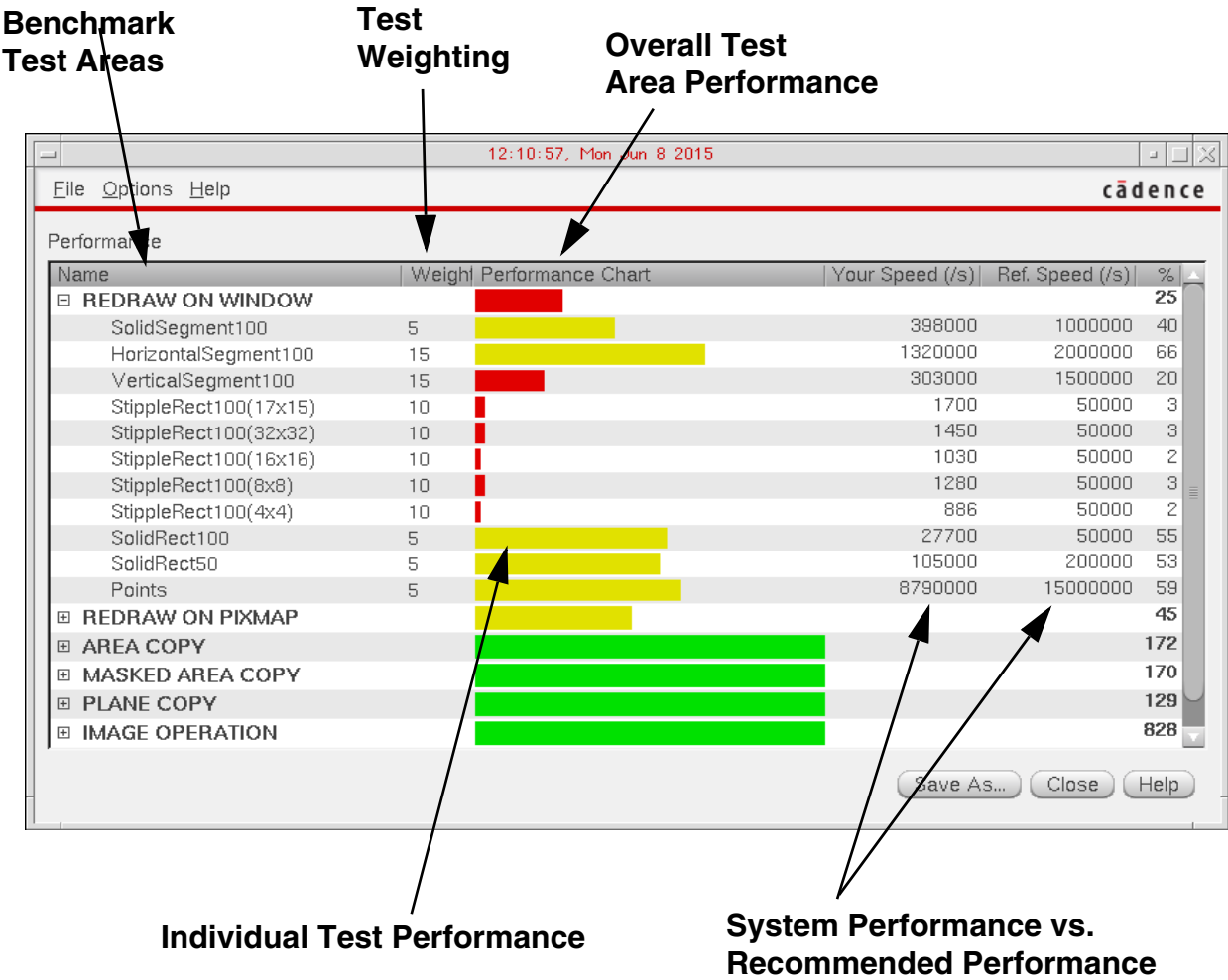
1. To run the benchmark tests that are currently selected in the Graphics Performance Benchmarks window, click the *Run* button, or choose *File – Run Selected Benchmarks*. A Testing window is displayed, and after the testing is complete, the results are displayed in the Benchmark Results window.
2. 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.



Virtuoso Studio Design Environment User Guide

Diagnostics



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 <i>Options</i> menu.
Weight	The weight of an individual benchmark that is used to measure its contribution to the performance of its benchmark group.

Virtuoso Studio Design Environment User Guide

Diagnostics

Column	Description
<i>Performance Chart</i>	<p>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.</p> <p>With the color scheme currently selected in the figure above, <i>Green</i>, <i>Yellow</i>, and <i>Red</i> are used to relay <i>Performance Chart</i> results. If required, you can change the color scheme used by choosing <i>Option – Color Code</i>.</p> <p>Additionally, you can use the <i>Options</i> menu to change the results display from a <i>2D Bar</i> to a <i>3D Bar Chart Presentation</i>, and also to choose whether to display <i>All Benchmarks</i> or <i>Only Selected Benchmarks</i>.</p> <p>After any changes have been made to the window settings, you can save them for reuse by choosing <i>Option – Remember Settings</i>.</p>
<i>Your Speed (/s)</i>	The absolute performance for a benchmark test.
<i>Ref Speed (/s)</i>	The collective performance that represents a reference standard that is deemed “good enough” for smooth performance.
<i>%</i>	The ratio of <i>Your Speed</i> against <i>Ref Speed</i> .

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*.

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.

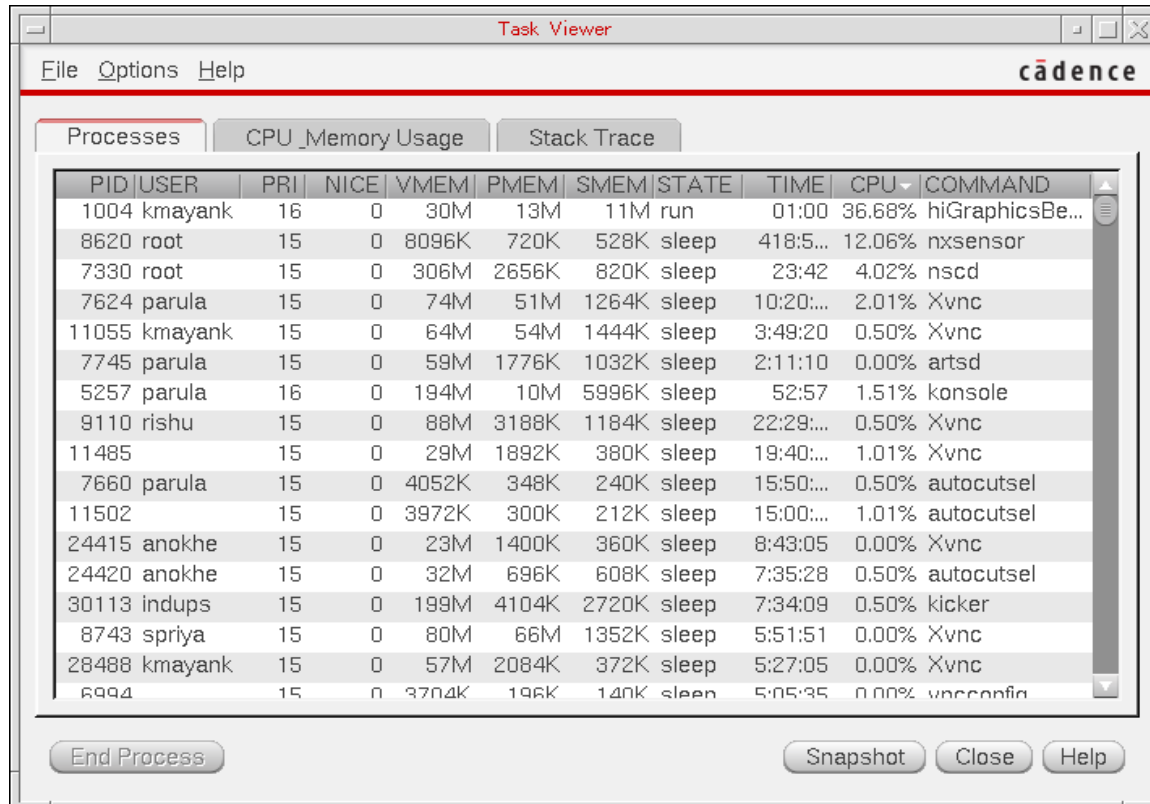
Related Topics

[Graphics Performance Benchmarks Window Form](#)

Task Viewer

In the Graphics Performance Benchmark window, choose *Tools – Task Viewer*, to display the Task Viewer window. The *Task Viewer* automatically updates after every 2 seconds.

Note: The Task Viewer tool is currently available only on Linux.



In the Task Viewer, you can view the *Processes* that are currently being run to determine which processes are going to impact the performance analysis results, and also the *CPU_Memory Usage*, *X Resources*, and the *Stack Trace*.

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.

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.

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.

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*.

Virtuoso Studio Design Environment User Guide

Diagnostics

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

Virtuoso Studio Design Environment User Guide

Diagnostics

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

Virtuoso Studio Design Environment User Guide

Diagnostics

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

Note: The Image operation tests how fast a client program such as the Virtuoso application can get a piece of image data from Xserver and send the data back after some manipulation. There is a round trip involved between the Virtuoso application and Xserver. As a result, if Xserver is far away from the client program, it can be very slow, especially for some thin-client platforms

Virtuoso Studio Design Environment User Guide

Diagnostics

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

Related Topics

[Graphics Performance Benchmarks Window Form](#)

Diagnostic Center Overview

The Diagnostic Center in the Virtuoso custom IC design platform is a one-stop shop for self-help and troubleshooting performance issues. It contains many debugging techniques to detect issues that might have caused an application to slow down.

Additionally, the Diagnostic Center provides an option to check your system and environment variables to identify potential setup issues, scan your libraries for data integrity issues, or gather other relevant data for debugging.

It has two tabs to perform these various checks,

- The *Performance* tab has the Health Monitor, which provides information to help you set up Virtuoso for better performance. This tab offers multiple checks to detect potential setup issues, such as checking environment variables, library data, design statistics, and SKILL profiling.
- The *Data Integrity* tab helps to detect data integrity issues by performing various checks, such as scanning libraries and hierarchies.

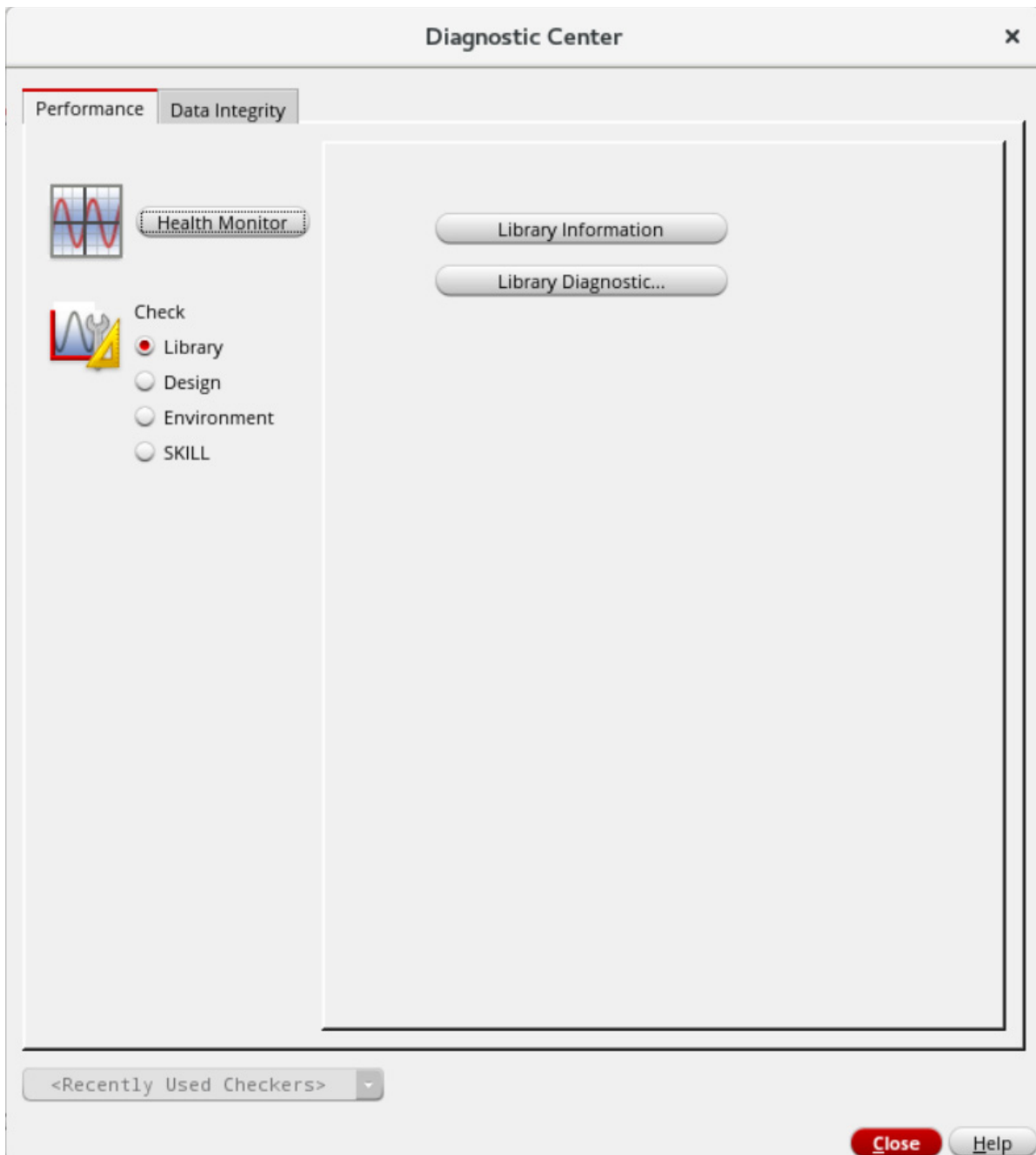
The Constraint Integrity checker and a subset of the Database Integrity checker are available in both tabs. It is because a data integrity issue might also appear as a performance issue, such as excessive unreferenced constraints or a recursive loop in the design hierarchy.

To open the Diagnostic Center, choose *Tools – Diagnostic Center* from the CIW menu bar.

Virtuoso Studio Design Environment User Guide

Diagnostics

The Diagnostic Center form appears.



Related Topics

[Diagnostic Center Form](#)

[Health Monitor Overview](#)

Health Monitor Overview

The Health Monitor in the Diagnostic Center provides information on the tool insight to help you set up the Virtuoso for better performance. It contains an indicator that turns yellow or red, with a performance tip when the current session slows down. It also monitors the system resources utilized by Virtuoso and other tools to identify potential external factors.

Additionally, this tool provides an option to record callstacks for reporting the slowness issue. These callstacks are then analyzed further and a high-level summary is created. If the Cadence support team is unable to reproduce the issue based on your problem description, this summary makes it easier for them 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 available in this tool.

A callstack is a snapshot of the internal function calls during program execution. A series of callstacks within the issue duration (profiling) can give the Cadence support team insights about where the most time is spent.

To open the Health Monitor form,

1. Open the *Tools* menu in CIW and choose *Diagnostic Center*.
2. In the Performance tab, select *Health Monitor*.

The Health Monitor form appears.



Virtuoso Studio Design Environment User Guide

Diagnostics

Alternatively, you can use the shell command `cdsPerfDiag -p <pid>` to open the Health Monitor when virtuoso stops responding or search `freeze` in `CDS.log` for this command. To open the Health Monitor at startup, set the `launchUI` environment variable.


This utility is installed by default through `perf installAtStartup` environment variable.

The `installAtStartup` environment variable is ignored when Virtuoso replays the command file, and this stops the installation of Health Monitor. If you want to install it when Virtuoso replays the command file, set the shell environment `setenv CDS_PERFDIAG_TEST`.

Set the shell environment `setenv CDS_PERFDIAG_DISABLE`, if the regression test is not based on replay commands and you want to disable the installation to prevent impacting the regression test.

From the Health Monitor form, you can access the Health Monitor (Advanced) form. This form provides performance insights and tips to debug the issues in the current Virtuoso session. It plots CPU and Memory charts consumed by Virtuoso and other processes, with historical system statistics to identify the external factors. You can also open the `CDS.log` augmented with events to debug the issues when the traffic light turns yellow or red over a long period.

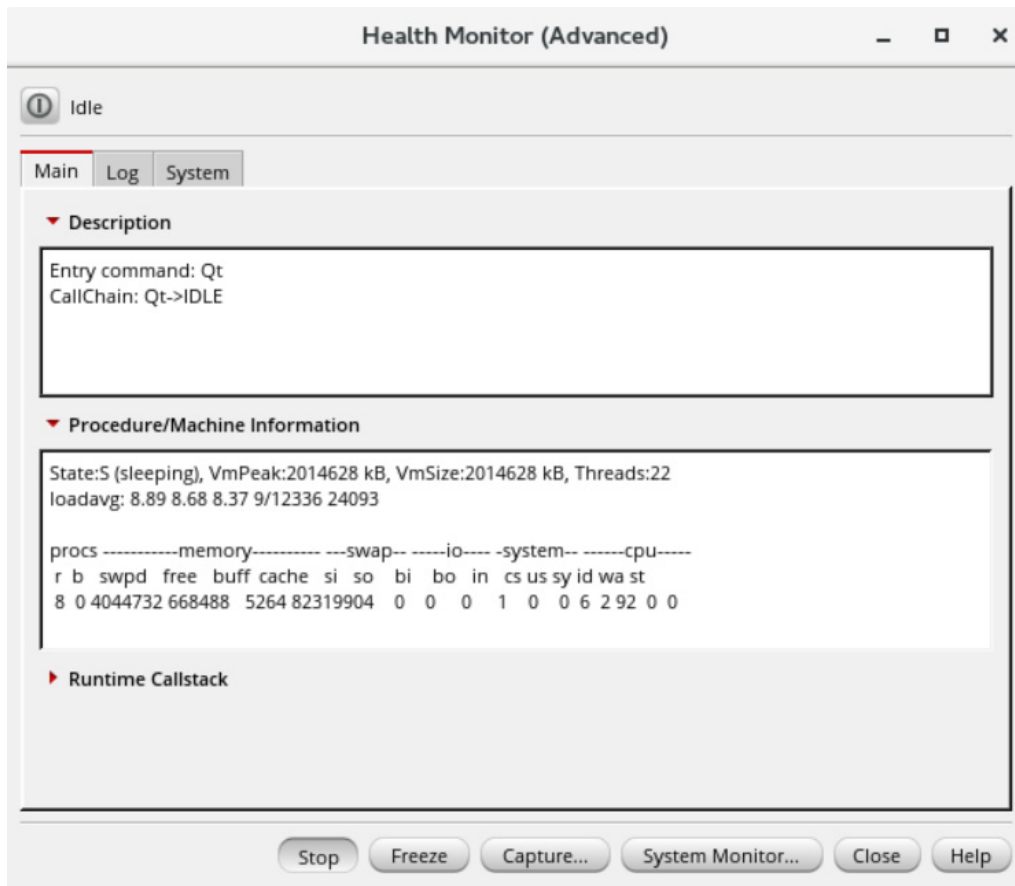
To open the Health Monitor (Advanced) form,

† Click the  icon on the Health Monitor form.

Virtuoso Studio Design Environment User Guide

Diagnostics

The Health Monitor (Advanced) form appears.



Related Topics

[Health Monitor Form](#)

[Health Monitor \(Advanced\) Form](#)

[launchUI](#)

[pinnedOffsetX](#)

[Collecting Data Using Health Monitor Tool](#)

[Diagnostic Center Form](#)

[Diagnostic Center Overview](#)

Collecting Data Using Health Monitor Tool

The Health Monitor 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 – Diagnostic Center*.
2. In the Performance tab, select *Health Monitor*.

The Health Monitor form appears.



3. Adjust the recording time.
4. Click *Start Recording* when the issue appears.
5. Click *Stop Recording* when you want to stop recording the callstacks.

Use the SKILL command `perfRefreshControl` to refresh the Health Monitor form and apply all the changes made using environment variables.

Related Topics

[Health Monitor Overview](#)

Reporting Performance Issues

For intermittent slowness, after you stop collecting data, a log file opens in the 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 the names of the functions on which the most time was spent. Functions with larger and smaller extremes are of particular interest. If the time taken is higher with a small depth then it might be the cause of bottleneck.

Simplified call-chains by package ($\geq 3\%$):

Order	Time(%)	Count	Call-chains
1	30.6	53	IL->CPH->OA->DB->ROD->TECH->SYSTEM
2	25.4	44	IL->CPH->OA->DB->ROD->TECH->OA

High-level summary on main thread: total 173 non-idle callstacks ('*' $>30\%$)

Depth	Time(%)	Count	Function
* 1	30.6	53	OpenAccess_4::oaDataTbl::read
* 2	31.2	54	OpenAccess_4::oaDatabase::readDataTbl
* 3	54.9	95	OpenAccess_4::oaDatabase::read

where,

- **Simplified call chain:** Groups function calls in a callstack using software packages, such as openAccess (OA) or tech file (TECH).
- **Time(%):** Displays approximated percentage of time spent in each function call. Closer to reality when more callstacks are recorded.
- **Depth:** Calculates the average depth measured from the top of each callstack. The function with a higher time percentage and a smaller depth can be a bottleneck.

In case an application freezes, Cadence recommends that you terminate the current Virtuoso session using the Health Monitor tool instead of killing it directly. Remember to enter your problem description in the Terminate Virtuoso form before you click the *Terminate Virtuoso* button. The Health Monitor tool collects the output of the UNIX `top` command and generates a high-level summary from 10 seconds of callstacks. This information is automatically appended to the `CDS.log` file and copied to `hangReport*.log.CDS.log`. You can send the hang report to the Cadence support team in the same way as reporting a crash.

Reporting Slowness

To report slowness using the Health Monitor tool, perform the following steps:

1. In the Health Monitor form, enable *Expert* mode.
2. Expand the *Program Execution Tracing* section.
3. Choose the *Open report form* option.
4. Click *Start Recording*.
5. After recording for some time, click *Stop Recording*.

The Report Slowness form appears.

6. Type the issues in the form and click *OK*.

This information is appended in the `slowReport*.log.CDS.log` with the system information and analyzed callstacks. Use the shell environment variable `CDS_ERRORLOG` to customize the file name and the log directory specified in the slow report.

To avoid opening the Report Slowness form and receive callstacks only, disable the *Open report form* option from the *Program Execution Tracing* section.

Related Topics

[Health Monitor Overview](#)

[Monitor by Strace Form](#)

[Health Monitor Form](#)

Troubleshooting an Unresponsive Virtuoso Application

To troubleshoot a sample application unresponsive:

1. Run the following SKILL loop command from CIW to make the application unresponsive:

```
while(t nil)
```

2. In the Health Monitor form, click *Start Recording*.
3. After recording for some time, click *Stop Recording*.
4. Click *Terminate Virtuoso*.

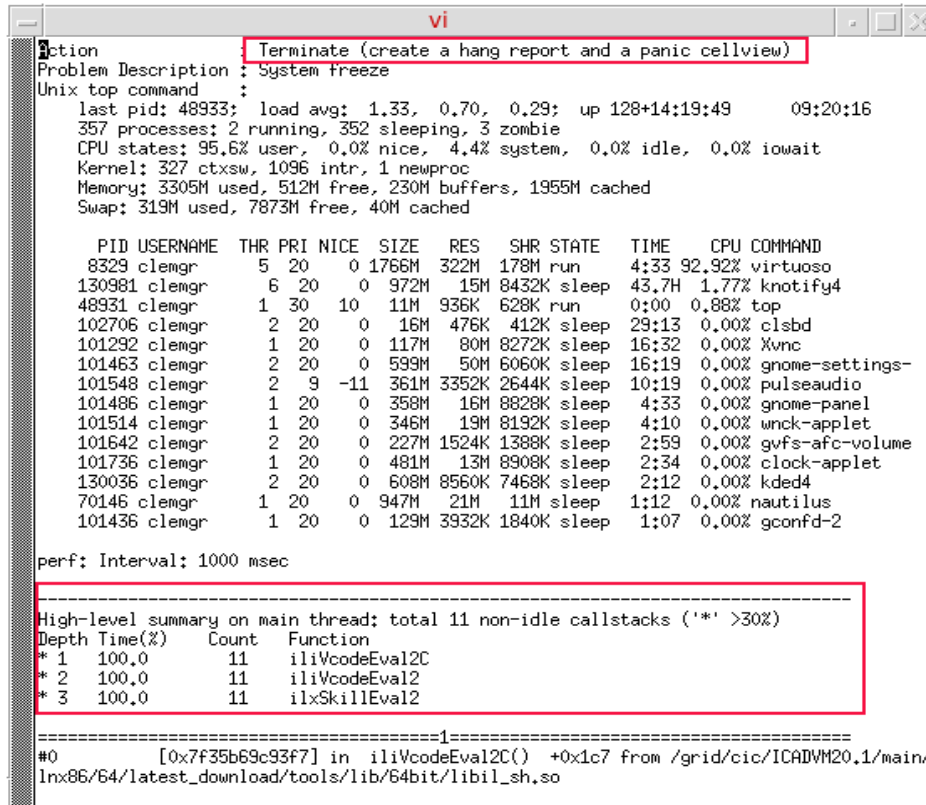
The Terminate Virtuoso form appears.

5. To break the SKILL loop, click the *Send CTRL+C* button to interrupt the application causing the issue.
6. Repeat step 1 to freeze the tool again.
7. Click *Terminate Virtuoso*.
8. Specify the reason for exiting Virtuoso (for example, describe how you run into this issue) and then click *Terminate*.

Virtuoso Studio Design Environment User Guide

Diagnostics

10 seconds of callstacks are collected and the Virtuoso session is terminated. In some cases, a hang report is also generated.



The screenshot shows a terminal window titled 'vi' with the following content:

```
Action: Terminate (create a hang report and a panic cellview)
Problem Description: System freeze
Unix top command:
  last pid: 48933; load avg: 1.33, 0.70, 0.29; up 128+14:19:49 09:20:16
  357 processes: 2 running, 352 sleeping, 3 zombie
  CPU states: 95.6% user, 0.0% nice, 4.4% system, 0.0% idle, 0.0% iowait
  Kernel: 327 ctxsw, 1096 intr, 1 newproc
  Memory: 3305M used, 512M free, 230M buffers, 1955M cached
  Swap: 319M used, 7873M free, 40M cached

  PID USERNAME THR PRI NICE SIZE RES SHR STATE TIME CPU COMMAND
  8329 clemgr 5 20 0 1766M 322M 178M run 4:33 92.92% virtuoso
  130981 clemgr 6 20 0 972M 15M 8432K sleep 43.7H 1.77% notify4
  48931 clemgr 1 30 10 11M 936K 628K run 0:00 0.88% top
  102706 clemgr 2 20 0 16M 476K 412K sleep 29:13 0.00% clsbd
  101292 clemgr 1 20 0 117M 80M 8272K sleep 16:32 0.00% Xvnc
  101463 clemgr 2 20 0 599M 50M 6060K sleep 16:19 0.00% gnome-settings-
  101548 clemgr 2 9 -11 361M 3352K 2644K sleep 10:19 0.00% pulseaudio
  101486 clemgr 1 20 0 358M 16M 8828K sleep 4:33 0.00% gnome-panel
  101514 clemgr 1 20 0 346M 19M 8192K sleep 4:10 0.00% wnck-applet
  101642 clemgr 2 20 0 227M 1524K 1388K sleep 2:59 0.00% gvfs-afc-volume
  101736 clemgr 1 20 0 481M 13M 8908K sleep 2:34 0.00% clock-applet
  130036 clemgr 2 20 0 608M 8560K 7468K sleep 2:12 0.00% kded4
  70146 clemgr 1 20 0 947M 21M 11M sleep 1:12 0.00% nautilus
  101436 clemgr 1 20 0 129M 3932K 1840K sleep 1:07 0.00% gconfd-2

perf: Interval: 1000 msec

High-level summary on main thread: total 11 non-idle callstacks ('*' >30%)
Depth Time(%) Count Function
* 1 100.0 11 iliVcodeEval2C
* 2 100.0 11 iliVcodeEval2
* 3 100.0 11 ilxSkillEval2

=====1=====
#0 [0x7f35b69c93f7] in iliVcodeEval2C() +0x1c7 from /grid/cic/ICADVM20.1/main/
lnx86/64/latest_download/tools/lib/64bit/libil_sh.so
```

You can conduct a thorough check of the design library access time with multiple pings to each library server to assess the average and worst case times. Depending on the number of libraries in the `cds.lib` file, this test might take several minutes.

1. To perform this test, in CIW, choose *Tools – Diagnostic Center*.
2. In the Performance tab, select *Check – Library*.

This option opens 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`

The same functionality is supported in *Library Manager – Help – Diagnostics*.

Checking the Profiler Summary

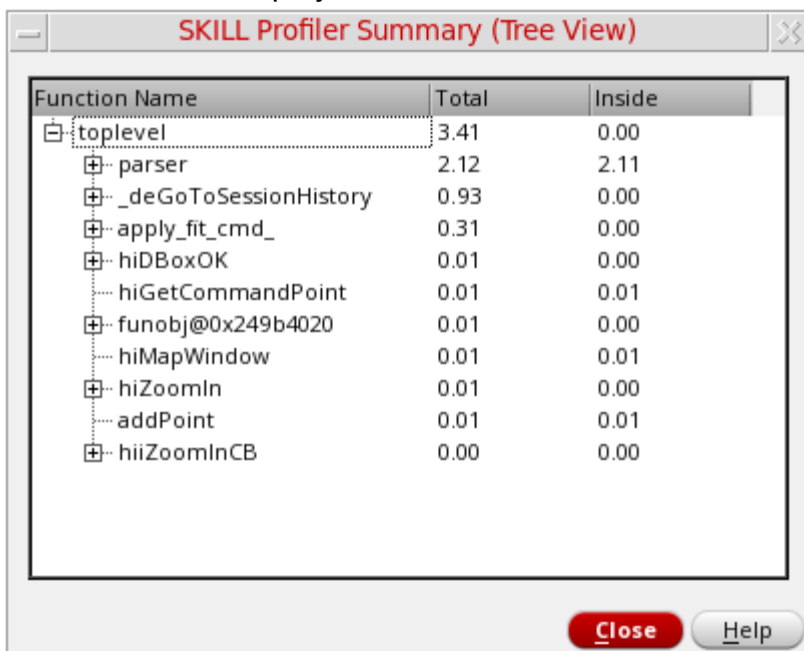
To view profiler summary in CIW:

1. Choose *Tools – Diagnostic Center*.
2. In the Performance tab, select *Check – SKILL – Profiler Summary*.

The Select a SKILL Profiler Summary Report form is displayed.

3. Select a saved profiler result file.

The results are displayed in a tree view.



Function Name	Total	Inside
[-] toplevel	3.41	0.00
[-] parser	2.12	2.11
[-] _deGoToSessionHistory	0.93	0.00
[-] apply_fit_cmd_	0.31	0.00
[-] hiDBoxOK	0.01	0.00
[-] hiGetCommandPoint	0.01	0.01
[-] funobj@0x249b4020	0.01	0.00
[-] hiMapWindow	0.01	0.01
[-] hiZoomIn	0.01	0.00
[-] addPoint	0.01	0.01
[-] hiiZoomInCB	0.00	0.00

Controlling the SKILL Replay

Interactively play a replay file from a `cdsmpls` 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.

1. To start, in CIW choose *Tools – Diagnostic Center*.
2. In the Performance tab, select *Check – SKILL – Replay*. Then follow the instructions in CIW.

Related Topics

[Additional Crash Data Collection](#)

Automatic Diagnostic Log Submissions in Virtuoso

In Virtuoso, a crash log is generated when an application fails or crashes. This crash log automatically records an abridged version of the crash data, including command logs and stack traces. The hang report is generated by the Health Monitor tool, when you terminate Virtuoso using the tool. The Health Monitor tool collects the output of the UNIX `top` command and generates a high-level summary from 10 seconds of callstacks. This information is automatically appended to the `CDS.log` file and copied to `hangReport*.log.CDS.log`.

This information can subsequently assist in the debugging of critical software problems and can also provide a means of tracking essential data that can be useful when attempting to identify key failure trends across specific applications (as well as Virtuoso itself).

Virtuoso automatically stores diagnostic logs (crash logs and hang logs) in a directory specified by `CDS_ERRORLOG`.

Virtuoso Analysis of Interaction Learning System (VAILS) provides the additional capability of automatically sending these logs to Cadence. When VAILS is enabled, diagnostic logs are:

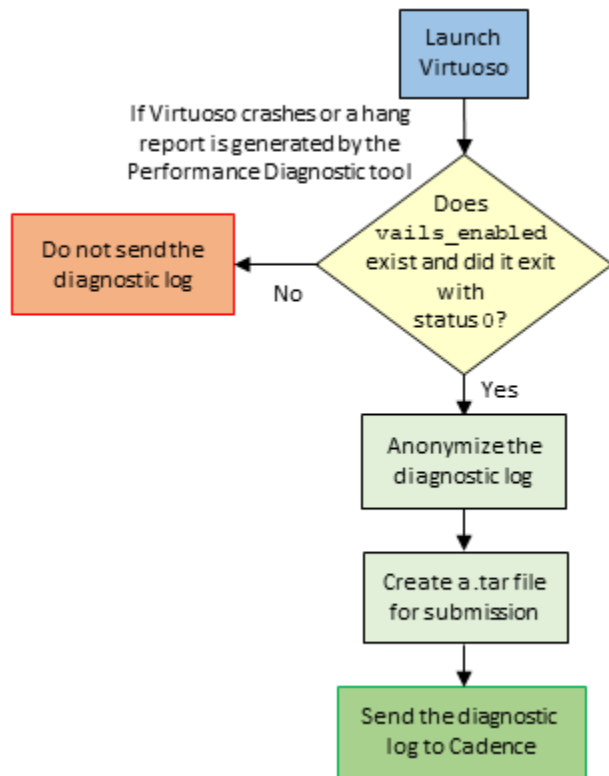
1. Automatically collected from your system.
2. Anonymized and made ready for submission to Cadence as a `.tar` file based on the criteria you set.
3. Sent to Cadence using the Amazon Web Services (AWS) Simple Storage Service (Amazon S3).

Note: The hosts on which Virtuoso is run must have access to AWS across the public internet to automatically send log files using VAILS.

Virtuoso Studio Design Environment User Guide

Diagnostics

The following flowchart depicts the VAILS flow:



Related Topics

[Health Monitor Overview](#)

[VAILS Enabled Script](#)

[Anonymization of Diagnostic Logs](#)

VAILS Enabled Script

VAILS is enabled when there is a script called `vails_enabled` present in the install hierarchy at the following location:

```
install_dir/share/cdssetup/vails/vails_enabled
```

A sample script called `vails_enabled.sample` is available in the `vails` folder. You can rename this script to `vails_enabled` to enable VAILS and start submitting diagnostic logs.

For the `vails_enabled` script to run successfully, it must not be empty. Additionally, it must be readable and executable. You can set the executable flag on the script by using the following command:

```
chmod a+rx vails_enabled
```

When this script is present, the `.VirtuosoMonitor` wrapper script is launched. This script triggers the automatic diagnostic log submission process when Virtuoso exits.

The `vails_enabled` script is passed two arguments:

- Submission type, which is a string with value "crash" or "hang".
- Path to directory where diagnostic logs are stored.

When Virtuoso exits, the `vails_enabled` script is called, and a decision is made about whether any diagnostic logs need to be sent to Cadence based on the criteria specified in the script.

For example, to send all diagnostic logs, the `vails_enabled` script might contain:

```
#!/bin/sh

# If the vails_enabled script file exists in the Cadence directory
# (<install_dir>/share/cdssetup/vails) VAILS will be enabled. It will be called
# before log submission and if it returns a failure exit status submission will
# be suppressed, otherwise it will proceed.

type="$1"
log="$2"

case $type in
    crash | hang)
        # Submit logs for all crashes and hangs
        exit 0
```

Virtuoso Studio Design Environment User Guide

Diagnostics

```
;;
*)
# By default, suppress all other submissions. In the future, other types of
# submission will be enabled, including for non-crashing sessions.
exit 1
;;
esac
```

You can further customize this script.

The following example shows how to set up the `vails_enabled` script to send a crash log only for users who have a file called `enable_vails` in their `HOME` directory. No crash logs are sent for other users.

```
if [ $type = "crash" ] && [ -f $HOME/enable_vails ]; then
    exit 0
fi
exit 1
```

This script does not submit the hang longs because `$type` is set to `crash`. To submit hang logs also use:

```
if ( [ $type = "crash" ] || [ $type = "hang" ] ) \
    && [ -f $HOME/vails_enable ]; then
    exit 0
fi
exit 1
```

You can further customize this script to add conditions and actions. For example, you can add a condition to filter the submissions based on a specific user, group, or project.

If the `vails_enabled` script does not exist, the diagnostic logs are not sent to Cadence automatically.

If the crash report gets submitted successfully, the following message is displayed in the terminal:

```
A diagnostic report has been submitted to Cadence for investigation.
```

If the submission fails, the following message is displayed in the terminal:

```
A diagnostic report was not submitted to Cadence because the
'vails_enabled' configuration script chose not to submit this error.
```

Related Topics

[Automatic Diagnostic Log Submissions in Virtuoso](#)

Anonymization of Diagnostic Logs

When diagnostic logs are sent to Cadence, you might want to anonymize the collected data to hide confidential information.

The following types of data can be removed, replaced, or anonymized (hashed) in the diagnostic logs:

- Personally identifiable data, such as user names, host names, and IP addresses.
- Customer-sensitive information, such as PDK and design names.

You define the rules used to anonymize data in a configuration file and place the file at the following location:

```
install_dir/share/cdssetup/vails/vailsAnonymizeLog.cfg
```

Note: The diagnostic logs are not submitted automatically if the `vailsAnonymizeLog.cfg` file is not available in the `vails` folder and cannot be parsed because conditions required for anonymization are not fulfilled.

The following is an example of an anonymization configuration file:

```
#####  
##  
# Rule name # Regular expression # Action # Grp #Replacement  
XDisplayName, ^.*X display name[:]*:\s*\([^ ]*\).*$, replace, 1, XDISPLAYNAME  
HostName, ^.*Host name[:]*:\s*\([^ ]*\).*$, replace, 1, HOSTNAME  
UserName, ^.*User Name:\s*\(.*)$, anonymize, 1
```

You can customize this configuration file based on your requirements. Each row in this file has the following fields:

- **Rule name:** Specifies the name of the rule.
- **Regular expression:** Describes the item that gets replaced. This expression uses the sed regular expression syntax.
- **Action:** Specifies which action to take when an expression matching the specified criteria is found. You can specify `remove`, `replace`, or `anonymize`.
- **Grp:** Specifies the name of the regular expression group to which the option specified in the **Action** field is applied.
- **Replacement:** Specifies the expression to replace the regular expression identified for replacement. This field is used only when the **Action** is set to `replace`.

Virtuoso Studio Design Environment User Guide

Diagnostics

The anonymization process is run prior to sending diagnostic logs to Cadence. By default, the `vailsAnonymizeLog.cfg` file shipped with the release anonymizes user, host, and display names.

In this process, the regular expressions are first applied only to the `CDS.log` file to identify the items to be anonymized. These actions are then applied to both the `CDS.log` and diagnostic log files to anonymize these items.

The following example shows how personal and confidential information in diagnostic logs is anonymized:

```
\o Program: @(#)CDS: virtuoso version 6.1.8-64b 09/09/2019 13:49 (ip-172-18-22-52) $
\# Host name (type): pc-samanthak (x86_64)
\# Operating system: Linux 2.6.32-431.11.2.el6.x86_64 #1 SMP Mon Mar 3 13:32:45 EST 2014
\# Linux /etc/issue: Red Hat Enterprise Linux Workstation release 6.5 (Santiago)
\# X display name (WdH): samanthak :1036 (1680x1005)
\# X version: 11.0 (vendor release 70000000)
\# System memory: 518,630 MB
\# Max mem available: 170,565 MB
\# User Name: samanthak
\o Working Directory: pc-samanthak :/grid/cic/eadrd_t2b_001/dev/samanthak/testcases/cloud-demo-mc
```

Specifies the anonymization rules to be applied.

```
#####
# Rule name # Regular expression # Action # Grp # Replacement
XDisplayName, ^.*X display name[^:]*:s*([^\ ]*).*$, replace, 1, XDISPLAYNAME
HostName, ^.*Host name[^:]*:s*([^\ ]*).*$, replace, 1, HOSTNAME
UserName, ^.*User Name:s*(.*)$, anonymize, 1
```

Confidential data is replaced and anonymized from the crash log to be sent to Cadence.

```
\o Program: @(#)CDS: virtuoso version 6.1.8-64b 09/09/2019 13:49 (ip-172-18-22-52) $
\# Host name (type): HOSTNAME (x86_64)
\# Operating system: Linux 2.6.32-431.11.2.el6.x86_64 #1 SMP Mon Mar 3 13:32:45 EST 2014
\# Linux /etc/issue: Red Hat Enterprise Linux Workstation release 6.5 (Santiago)
\# X display name (WdH): XDISPLAYNAME (1680x1005)
\# X version: 11.0 (vendor release 70000000)
\# System memory: 518,630 MB
\# Max mem available: 170,565 MB
\# User Name: b03735e76fcd64be2ff3d0472c549ec8dca07c6d
\o Working Directory: HOSTNAME:/grid/cic/eadrd_t2b_001/dev/b03735e76fcd64be2ff3d0472c549ec8dca07c6d/testcase
s/cloud-demo-mc
```

Related Topics

Conditions for Anonymization

Use of UNIX 'sed' Command in Anonymization

The UNIX 'sed' command helps to process the log files. The `Regular expression` in the configuration file is passed to sed, which, in turn, identifies the string that needs to be replaced or removed. If the string is unknown, for example the login name of a user, then you need to search for the string `User Name` and then extract the name of the user as the string to be replaced.

The process has two steps:

- Identifying the complete string including the identifying context. This is achieved by using the `Regular expression` field.
- Extracting just the substring that needs to be replaced or removed. This is achieved by using the `Grp` field.

Example 1

Consider the following line of the configuration file, which anonymizes the username.

```
UserName,      ^.*User Name:\s*\(.*\)$,      anonymize, 1
```

Here, the `Regular expression` is `^.*User Name:\s*\(.*\)$` and `Grp` is 1.

The line in the log file is:

```
\# User Name:      samanthak
```

In the `Regular expression`:

- `^`: Matches the beginning of the line in the log file
- `.*`: Matches the zero or more arbitrary characters before the string `User Name`: that is, the string `\#`.
- `User Name:` Matches the string `User Name:`.
- `\s*`: Matches the whitespace between `User Name:` and the name of the user. For example, `samanthak`.
- `\(.*\)`: The expression `.*` matches an arbitrary sequence of characters, such as `samanthak`. Since this value is referenced later, it requires escaped parenthesis. Each pair of the parentheses is assigned an increasing number. In this example, only one set is specified, so it is assigned the value 1.

The `Grp` is set to 1 to indicate that this is the value that needs to be replaced.

Example 2

In this example, consider the same regular expression in multiple rules to identify and anonymize schematic library and cell names. For this, we can use a general replay command of the form `function("libName" "cellName" "schematic")` to identify the library and cell names. For example, the replay command for opening a cellview from the history list is:

```
\i cv = dbOpenCellViewByType("opamp090" "full_diff_opamp_AC" "schematic")
```

Now, create a regular expression to identify the argument list. So, for the following argument list:

```
"opamp090" "full_diff_opamp_AC" "schematic"
```

The regular expression is as follows:

```
^.*"\[([^\"]*)\]"\\s\\+\"\[([^\"]*)\]"\\s\\+"schematic".*$
```

In this expression, the first two values in the list are used to identify the library and cell names.

The components of the regular expression are as follows:

- `^. *"`: Matches everything from the beginning of the line up to the first quote.
- `\\([^\"]*)\\`: Matches the string inside the quotes, which can be referenced as `Grp 1`.
- `\\s\\+`: Matches one or more whitespace characters.
- `"`: Matches the opening and closing quote of the second string.
- `\\([^\"]*)\\`: Matches the string inside the second set of quotes, which can be referenced as `Grp 2`.
- `"schematic"`: Matches the literal string `"schematic"`.
- `.*$`: Matches the remainder of the line.

We can use this expression to create the following rules. These rules use the same regular expression, but use a different `Grp` to identify the value to anonymize:

- Anonymizes the library name

```
^.*"\[([^\"]*)\]"\\s\\+\"\[([^\"]*)\]"\\s\\+"schematic".*$, anonymize, 1
```

This rule extracts and anonymizes the library name `opamp090`.

- Anonymizes the cell name.

```
^.*"\[([^\"]*)\]"\\s\\+\"\[([^\"]*)\]"\\s\\+"schematic".*$, anonymize, 2
```

This rule extracts and anonymizes the cell name `full_diff_opamp_AC`.

Conditions for Anonymization

For anonymization to work, certain conditions must be met. If anonymization fails, diagnostic logs are not submitted automatically.

For anonymization to work, ensure that:

- The `/bin/awk`, `/bin/sed`, `/usr/bin/id`, and `/usr/bin/openssl` files exist and are executable. These files are needed to perform anonymization.
- The `CDS.log` or `vailsAnonymizeLog.cfg` file exists and is readable.
- There are no syntax issues in the `vailsAnonymizeLog.cfg` file. This means that:
 - ❑ The `Regular expression`, `Action`, or `Group` fields should not be blank or missing.
 - ❑ If `replace` is specified in the `Action` field, then the `Replacement` field must exist and must not be blank.
 - ❑ The `Action` field should have one of the following values: `anonymize`, `replace`, or `remove`.

Related Topics

[Use of UNIX 'sed' Command in Anonymization](#)

Validation of Automatic Diagnostic Log Submission

You can validate that the automatic diagnostic log submission is working with the help of the `vails_test` script. This script is located in the following folder:

```
install_dir/share/cdssetup/vails/
```

By default, this script checks the whether the `vails_enabled` script is readable and executable.

To run the `vails_test` script, type the following command at the UNIX prompt and press Enter:

```
> $ $share/cdssetup/vails/vails_test
```

If the `vails_enabled` script is working, the following messages are displayed in the output:

```
[OK] Located root of virtuoso installation at /servers/cic_pe/
asanwal/hier/ICADVM20.1ISR/latest_download
```

```
[OK] '/servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/
share/cdssetup/vails/vails_enabled' script is installed, and is
readable and executable.
```

```
[OK] 'vails_enabled' script allows log upload after crash
```

```
[OK] 'vails_enabled' script allows log upload after hang
```

```
[OK] 'vails_enabled' script disallows log upload for event type other
than 'crash' and 'hang'.
```

```
[INFO] Checking if AWS is reachable...
```

```
[INFO] Received response.
```

```
[INFO] AWS is reachable.
```

```
[OK] VAILS configuration is good.
```

If there are any errors or warnings, they are reported in the following way:

```
[WARN] 'vails_enabled' script does not allow log upload after crash.
If this is expected, ignore this warning.
```

```
[WARN] 'vails_enabled' script does not allow log upload after hang.
If this is expected, ignore this warning.
```

Virtuoso Studio Design Environment User Guide

Diagnostics

```
[ERROR] 'vails_enabled' script allows log upload for event types
other than 'crash' and 'hang' . Other event types may be used in
future releases to trigger log submission. Check the documentation to
see how to modify the script to restrict log submission to crashes
and hangs.
```

```
[ERROR] 'vails_enabled' script is not executable.
```

```
[ERROR] 'vails_enabled' script is not readable.
```

```
[ERROR] 'vails_enabled' script is empty.
```

The `vails_test` script also checks whether AWS is reachable. This check is performed after validating the configuration of the `vails_enabled` script and before the submission of the test crash log. AWS is considered unreachable when there is no response received from it within 60 seconds.

When AWS is not reachable, the following messages are displayed:

```
[INFO] Checking if AWS is reachable...
```

```
[ERROR] Request failed with response:
```

```
Connection closed
```

```
[ERROR] AWS is not reachable from this machine. This could be due to
a firewall. If you have access to an HTTP proxy, try setting the shell
environment variable https_proxy to ":>http://:". 
```

```
[ERROR] Cannot submit logs because AWS is not reachable from eu-lnx-
38.
```

You can use the `vails_test` script for the following tasks:

Case 1: If you provide an additional argument `yes`, the script generates an artificial crash by issuing a fatal command in the CIW. If the `vails_enabled` script is working, the crash log gets created and submitted.

To run the `vails_test` script with argument `yes`, type the following command at the UNIX prompt and press Enter:

```
> $share/cdssetup/vails/vails_test yes
```

If the `vails_enabled` script is working, the following messages are displayed in the output:

Virtuoso Studio Design Environment User Guide

Diagnostics

[OK] Located root of virtuoso installation at /servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download

[OK] '/servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/share/cdssetup/vails/vails_enabled' script is installed, and is readable and executable.

[OK] 'vails_enabled' script allows log upload after crash

[OK] 'vails_enabled' script allows log upload after hang

[OK] 'vails_enabled' script disallows log upload for event type other than 'crash' and 'hang'.

[INFO] Checking if AWS is reachable...

[INFO] Received response.

[INFO] AWS is reachable.

[INFO] Preparing artificial virtuoso crash log for submission...

[INFO] Looking for path to crash report in /servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/share/cdssetup/vails/artificial_crash.log.

[INFO] Crash report file is /tmp/crashReport_060123_160424_ICADVM20.1-64b.500.33.EA7_asanwal_noivl-asanwal.log.

[INFO] Log type is crash.

[INFO] Target name is /tmp/anonymized_crashReport_060123_160424_ICADVM20.1-64b.500.33.EA7.log.

[INFO] Anonymization tool is /servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/tools/dfII/bin/vailsAnonymizeLog.sh.

[INFO] Command to anonymize file(s):

```
/servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/tools/dfII/bin/vailsAnonymizeLog.sh /tmp/crashReport_060123_160424_ICADVM20.1-64b.500.33.EA7_asanwal_noivl-asanwal.log.CDS.log /tmp/anonymized_crashReport_060123_160424_ICADVM20.1-64b.500.33.EA7.log.CDS.log /tmp/
```

Virtuoso Studio Design Environment User Guide

Diagnostics

```
crashReport_060123_160424_ICADVM20.1-64b.500.33.EA7_asanwal_noivl-  
asanwal.log /tmp/anonymized_crashReport_060123_160424_ICADVM20.1-  
64b.500.33.EA7.log
```

```
[INFO] Command to create archive file 103426.tar:
```

```
tar cf 103426.tar -C /tmp  
anonymized_crashReport_060123_160424_ICADVM20.1-  
64b.500.33.EA7.log.CDS.log -C /tmp  
anonymized_crashReport_060123_160424_ICADVM20.1-64b.500.33.EA7.log
```

```
[INFO] Command to compress archive file:
```

```
gzip 103426.tar
```

```
[INFO] Requesting URL for upload.
```

```
[INFO] Sending request.
```

```
[INFO] Received response.
```

```
[INFO] Received valid URL.
```

```
[INFO] Sending upload request.
```

```
[INFO] Received response.
```

```
[INFO] Successfully uploaded file.
```

```
[OK] Crash log submitted successfully.
```

```
[OK] VAILS configuration is good.
```

Virtuoso is launched, when Preparing artificial virtuoso crash log for uploading... is displayed in the output, and the CIW appears briefly before an artificial crash is invoked using a replay file.

Case 2: If the `vailsAnonymizeLog.cfg` file is missing. In this case, when you run the `vails_test` script with argument `yes` at the UNIX prompt, the following messages are displayed.

```
[OK] Located root of virtuoso installation at /servers/cic_pe/  
asanwal/hier/ICADVM20.1ISR/latest_download
```

Virtuoso Studio Design Environment User Guide

Diagnostics

```
[OK] '/servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/
share/cdssetup/vails/vails_enabled' script is installed, and is
readable and executable.

[OK] 'vails_enabled' script allows log upload after crash

[OK] 'vails_enabled' script allows log upload after hang

[OK] 'vails_enabled' script disallows log upload for event type other
than 'crash' and 'hang'.

[INFO] Checking if AWS is reachable...

[INFO] Received response.

[INFO] AWS is reachable.

[INFO] Preparing artificial virtuoso crash log for submission...

[INFO] Looking for path to crash report in /servers/cic_pe/asanwal/
hier/ICADVM20.1ISR/latest_download/share/cdssetup/vails/
artificial_crash.log.

[INFO] Crash report file is /tmp/
crashReport_060123_161108_ICADVM20.1-64b.500.33.EA7_asanwal_noivl-
asanwal.log.

[INFO] Log type is crash.

[INFO] Target name is /tmp/
anonymized_crashReport_060123_161108_ICADVM20.1-64b.500.33.EA7.log.

[INFO] Anonymization tool is /servers/cic_pe/asanwal/hier/
ICADVM20.1ISR/latest_download/tools/dfII/bin/vailsAnonymizeLog.sh.

[INFO] Command to anonymize file(s):

/servers/cic_pe/asanwal/hier/ICADVM20.1ISR/latest_download/tools/
dfII/bin/vailsAnonymizeLog.sh /tmp/
crashReport_060123_161108_ICADVM20.1-64b.500.33.EA7_asanwal_noivl-
asanwal.log.CDS.log /tmp/
anonymized_crashReport_060123_161108_ICADVM20.1-
64b.500.33.EA7.log.CDS.log /tmp/
crashReport_060123_161108_ICADVM20.1-64b.500.33.EA7_asanwal_noivl-
```

Virtuoso Studio Design Environment User Guide

Diagnostics

```
asanwal.log /tmp/anonymized_crashReport_060123_161108_ICADVM20.1-64b.500.33.EA7.log
```

```
[ERROR] Failed to anonymize file(s):
```

```
[ERROR] Unable to generate archive file.
```

```
[ERROR] Failed to submit crash log.
```

The script also returns an exit status that can be used to check the results when it is called from your own script. In case there are no errors exit status is zero. This means that all checks are fine, or all checks are fine with warnings, and a non-zero exit status if any error is reported.

vails_test Script to Launch Virtuoso using Job Scheduler

You can use the `CDS_VAILS_TEST_LAUNCH_CMD` shell environment variable to submit Virtuoso to a job scheduler rather than running it on the local machine. For example, the following command causes Virtuoso to be run by submitting a job to the `lnx64` queue on the Load Sharing Facility (LSF) cluster:

```
CDS_VAILS_TEST_LAUNCH_CMD 'bsub -K -q lnx64'
```

The `-K` flag is needed because the `vails_test` script cannot proceed with its checks until Virtuoso exits. `-K` causes the `bsub` (command that submits jobs to LSF) to wait until the job finishes.

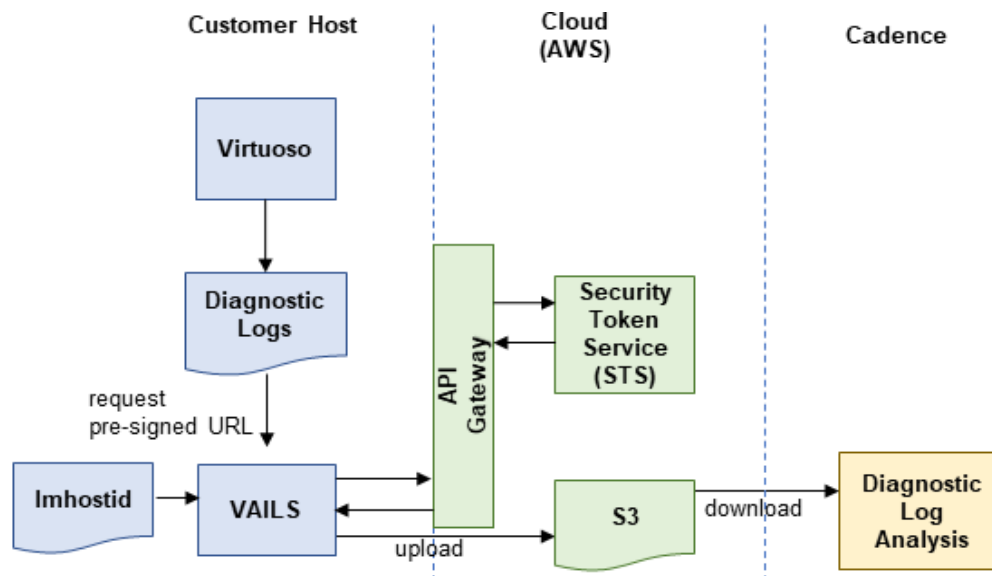
Related Topics

[VAILS Enabled Script](#)

How to Send the Diagnostic Logs to Cadence

Diagnostic logs are sent to Cadence using the cloud storage service Amazon S3, which ensures that only Cadence is able to see the contents of these diagnostic log files.

Using AWS for automatic diagnostic submission ensures that the storage server is always available to accept submissions. All submissions are secure because file transfers to and from the cloud are performed over a transport layer security (TLS) connection. Diagnostic logs are stored in the cloud using AWS - Key Management Service (KMS) server-side encryption.



To summarize, for automatic submission of diagnostic logs using VAILS, you need to ensure that:

- Your systems have access to AWS.
- The `vails_enabled` script exists and is executable.
- Anonymization rules have been configured based your requirements by modifying the `vailsAnonymizeLog.cfg` file.

Related Topics

[Automatic Diagnostic Log Submissions in Virtuoso](#)

[Anonymization of Diagnostic Logs](#)

Virtuoso Studio Design Environment User Guide

Diagnostics

oaScan Utilities for Virtuoso

oaScan is an unlicensed application that scans the contents of a library. It checks for inconsistencies in the OpenAccess design, technology, DMDData databases, and optionally repairs the inconsistencies and saves the databases.

Inconsistencies are more common with older versions of OpenAccess. 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

It is recommended 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.

You can load these three GUI-based utilities in Virtuoso and access from the Command Interpreter Window (CIW) to implement the recommended methodology.

- Scan/Repair Library: Runs oaScan on a library or cellview
- Scan/Repair Hierarchy: Runs oaScan on all the cellviews in a design hierarchy
- Scan on Save: Runs oaScan every time a cellview is saved

Related Topics

[Scanning a Library](#)

Scanning a Hierarchy

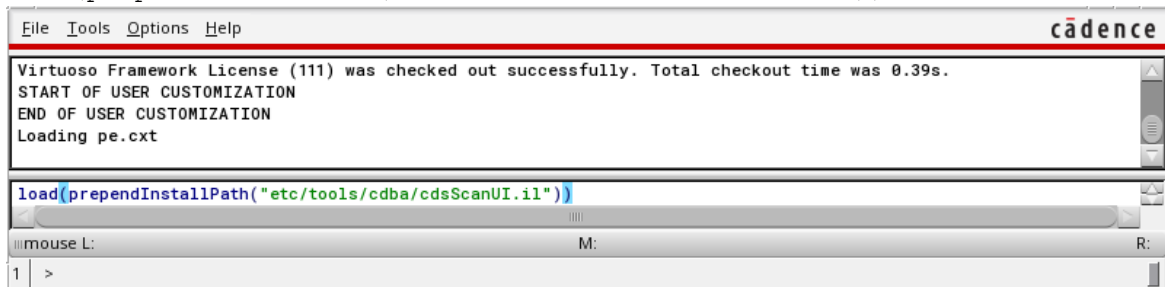
Scanning Cellviews Automatically During Save

Enabling the oaScan Utilities and Specifying the oaScan Version to Use

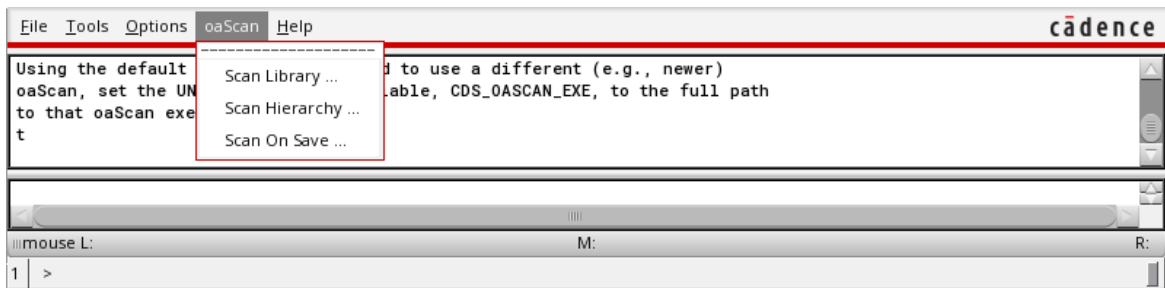
By default, the oaScan utilities are hidden in Virtuoso. To enable them:

- ➔ 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
```

Related Topics

Scanning a Library

Scanning a Hierarchy

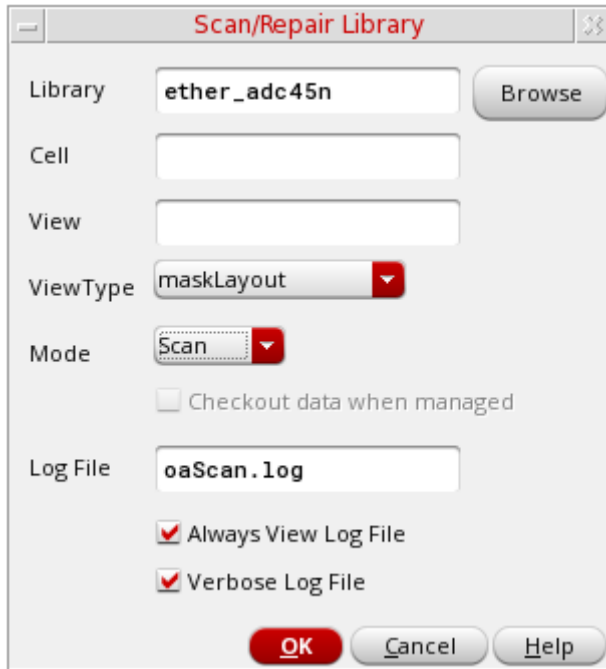
Scanning Cellviews Automatically During Save

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.

Related Topics

[Scan/Repair Library Form](#)

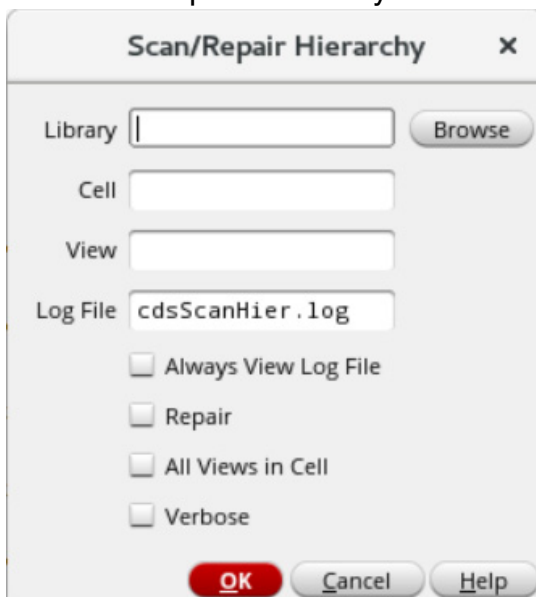
Scanning a Hierarchy

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.

The image shows a dialog box titled "Scan/Repair Hierarchy" with a close button (X) in the top right corner. The dialog contains several input fields and checkboxes. The "Library" field is empty, followed by a "Browse" button. The "Cell" and "View" fields are also empty. The "Log File" field contains the text "cdsScanHier.log". Below these fields are four unchecked checkboxes: "Always View Log File", "Repair", "All Views in Cell", and "Verbose". At the bottom of the dialog are three buttons: "OK" (highlighted in red), "Cancel", and "Help".

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 the view-file window on completion of the scan even if there are no issues found.
5. Check the *Repair* box to automatically repair any issues found in the design hierarchy.
6. Check the *All Views in Cell* box to scan all the views present in the specified cell. If this option is not selected, only the cellviews referenced in the design hierarchy are scanned.
7. Check the *Verbose* box to view the name of the databases that oaScan is processing.
8. Click *OK* to scan the hierarchy.

Related Topics

[Scan/Repair Hierarchy Form](#)

Scanning Cellviews Automatically During Save

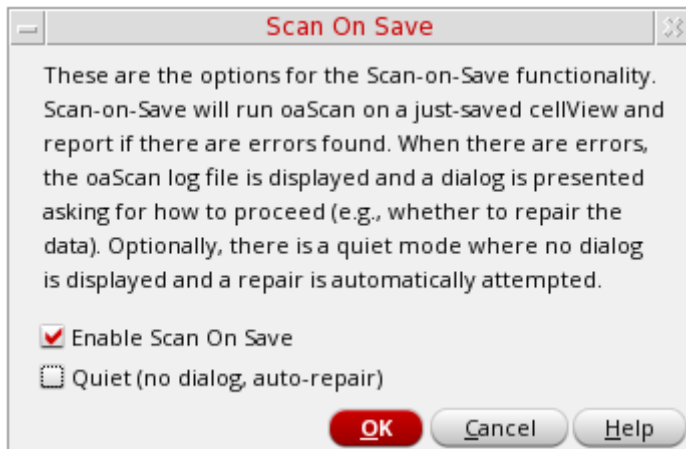
Instead of manually calling oaScan after each design update, you can run the check automatically any time you save data to disk. The software first makes a backup copy of the pre-save database and then runs oaScan on the newly-saved data. If a data issue has been introduced in the session since the last 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.

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.

Virtuoso Studio Design Environment User Guide

oaScan Utilities for Virtuoso

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:

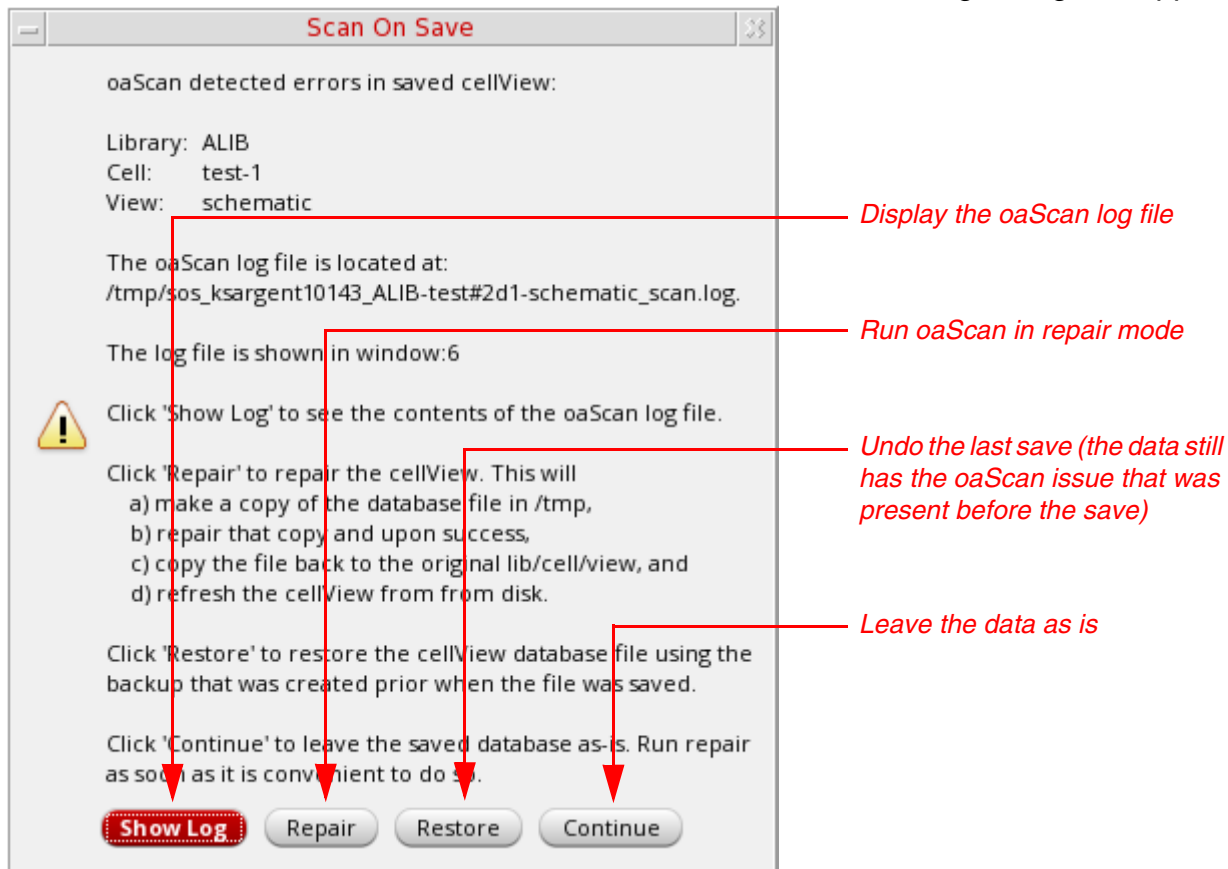
```
setenv CDS_SCANLOG "DIR=/home/user/scanlogs"
```

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.

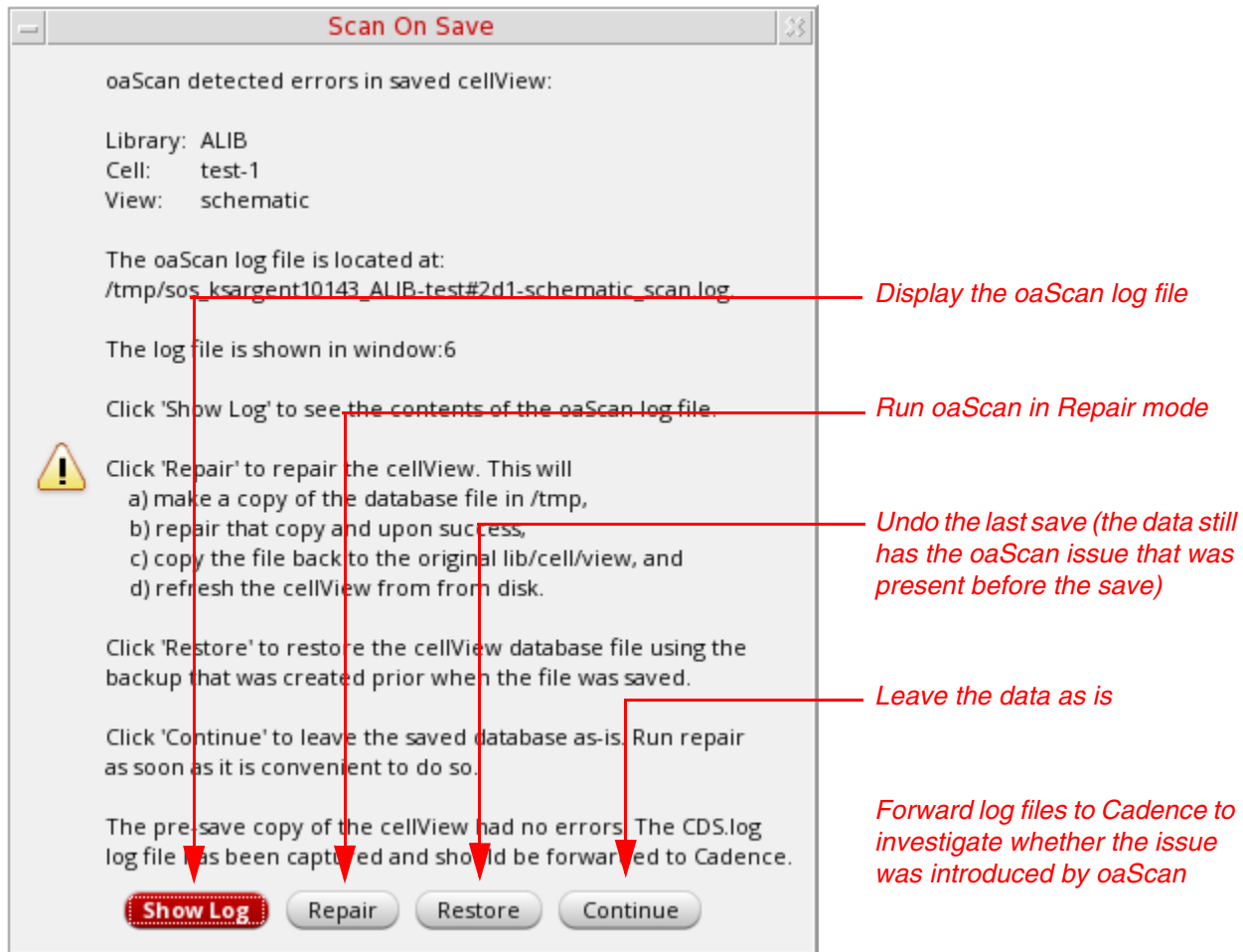
If the data contained an oaScan issue before the save. The following dialog box appears:



Virtuoso Studio Design Environment User Guide

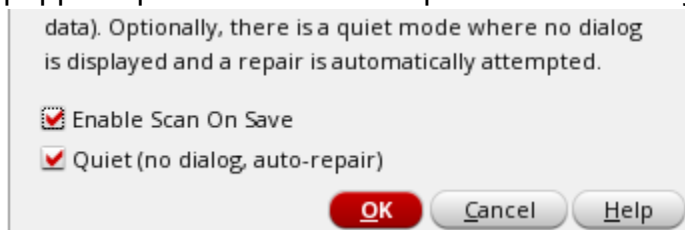
oaScan Utilities for Virtuoso

If an oaScan issue was found when the original data was clean. The following dialog box appears::



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.



Related Topics

[Scan On Save Form](#)

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.